ISSN 0265-2919
80p 25

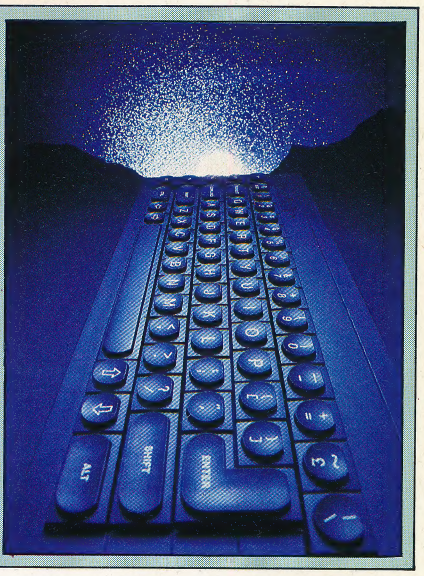
THE HOME COMPUTER ADVANCED COURSE

MAKING THE MOST OF YOUR MICRO

An ORBIS Publication
IR £1 Aus $1.95 NZ $2.25 SA R1.95 Sing $4.50 USA & Can $1.95

CONTENTS

Next Week



- Sinclair's QL has attracted exaggerated praise and criticism — we take a cool look at the machine, the makers' claims and the critics' jibes.
- LOGO users think it the ideal computing language. In the first instalment of a LOGO series we introduce the language's philosophy and history.
- The return of our popular Workshop series is prefaced next week by an explanation of the new series' technical background.

QUIZ

1) What are the parameters of the Fillsub machine code subroutine?

2) What is MUSICOMP, and who designed it?

3) Who is the central character of Lords Of Midnight?

4) What is the technique called 'firewalling'?

Answers To Last Week's Quiz

A1) A program is disassembled to ensure that the machine code program has been assembled as intended by the programmer.

A2) 'Top-down design' divides a program into a 'pyramid' of modules, making it easier to understand and debug.

A3) The Sony $3\frac{1}{2}$ in disk is used on the Apple Macintosh.

A4) '!' is known as the factorial function. 4!, for example, is equal to $4 \times 3 \times 2 \times 1 = 24$.

QUIZ

Editor Jim Lennox; **Managing Editor** Mike Wesley; **Art Director** David Whelan; **Technical Editor** Brian Morris; **Production Editor** Catherine Cardwell; **Art Editor** Claudia Zeff; **Chief Sub Editor** Robert Pickering; **Designer** Julian Dorr; **Art Assistant** Liz Dixon; **Editorial Assistant** Stephen Malone; **Sub Editor** Steve Mann; **Researcher** Melanie Davis; **Contributors** Steve Colwill, Steve Malone, Rory Forsyth, Geoff Nairn, Jim Lennox, Richard Pawson, Graham Storrs; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Chris Cooper; **Production Controller** Peter Taylor-Medhurst; **Circulation Director** David Breed; **Marketing Director** Michael Joyce; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 14 Rathbone Place, London W1P 1DE; **© APSIF Copenhagen 1984; © Orbis Publishing Ltd 1984: Typeset by Universe; Reproduction by Mullis Morgan Ltd; Printed in Great Britain by Artisan Press Ltd, Leicester**

HOME COMPUTER ADVANCED COURSE – Price UK 80p IR £1.00 AUS $1.95 NZ $2.25 SA R1.95 SINGAPORE $4.50 USA and CANADA $1.95

How to obtain your copies of HOME COMPUTER ADVANCED COURSE – Copies are obtainable by placing a regular order at your newsagent, or by taking out a subscription. Subscription rates: for six months (26 issues) £23.80; for one year (52 issues) £47.60. Send your order and remittance to Punch Subscription Services, Watling Street, Bletchley, Milton Keynes, Bucks MK2 2BW, being sure to state the number of the first issue required.

Back Numbers UK and Eire – Back numbers are obtainable from your newsagent or from HOME COMPUTER ADVANCED COURSE. Back numbers, Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT at cover price. AUSTRALIA: Back numbers are obtainable from HOME COMPUTER ADVANCED COURSE. Back numbers, Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G Melbourne, Vic 3001. SOUTH AFRICA, NEW ZEALAND, EUROPE & MALTA: Back numbers are available at cover price from your newsagent. In case of difficulty write to the address in your country given for binders. South African readers should add sales tax.

How to obtain binders for HOME COMPUTER ADVANCED COURSE – UK and Eire: Please send £3.95 per binder if you do not wish to take advantage of our special offer detailed in Issues 5, 6 and 7. EUROPE: Write with remittance of £5.00 per binder (incl. p&p) payable to Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT. MALTA: Binders are obtainable through your local newsagent price £3.95. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Miller (Malta) Ltd, M.A. Vassalli Street, Valletta, Malta. AUSTRALIA: For details of how to obtain your binders see inserts in early issues or write to HOME COMPUTER ADVANCED COURSE BINDERS, First Post Pty Ltd, 23 Chandos Street, St. Leonards, NSW 2065. The binders supplied are those illustrated in the magazine. NEW ZEALAND: Binders are available through your local newsagent or from HOME COMPUTER ADVANCED COURSE BINDERS, Gordon & Gotch (NZ) Ltd, PO Box 1595, Wellington. SOUTH AFRICA: Binders are available through any branch of Central Newsagency. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Intermag, PO Box 57394, Springfield 2137.

Note – Binders and back numbers are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK market only and may not necessarily be identical to binders produced for sale outside the UK. Binders and issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.

COVER PHOTOGRAPHY BY MARCUS WILSON-SMITH, MUSIC STAND COURTESY OF THOMAS JERRAMS ANTIQUES

THE MELODY MAKERS

This is the first in a series of articles in which we will be looking in some detail at MIDI — the Musical Instrument Digital Interface. We will also discuss how digital manipulation of sound — through sequencing, modulation synthesis and the sampling of natural sounds — can produce results hardly imaginable a decade ago.

Music and the sciences of number and proportion have a relationship that has been acknowledged for many centuries. The Greek mathematician Pythagoras weighed a group of blacksmiths' hammers to find out why they seemed to be playing 'in tune' as the anvil was struck. He found that one hammer, half the weight of another, produced a sound exactly double the frequency, or one octave higher. This established the first principle governing pitch relationships in music.

In the Middle Ages, composers filled cathedrals with the sound of masses and motets (polyphonic choral compositions) that were rhythmically and numerically proportioned to the same degree of precision as the architecture of the cathedrals themselves. Their music was often so complex that it was believed that only the ears of God could appreciate the numerical relationships, whilst mere humans heard the music. And anyone who has watched a performance of live music — of almost any variety — may have noticed musicians counting, '1,2,3,4; 2,2,3,4' under their breath, before they start to play.

So it was natural that the worlds of computing and music would overlap, and at present a development causing a lot of excitement is MIDI — the Musical Instrument Digital Interface. This unit is designed to enable any one digital system, including microcomputers, to control the functions of another. As the majority of electronic musical instruments now being produced are digital, this opens up a whole new realm of exciting possibilities to home micro owners.

But MIDI is not a magic box. It will not turn a micro owner into a Vangelis or a Stevie Wonder overnight. Musical skills and imagination will always produce the best results, whether the music is being played on a bank of interfaced synthesisers or on an acoustic guitar.

In order to understand the sort of musical instruments with which MIDI is intended to interface, and how electronic music came about, we have to look back over half a century. Well before the Second World War, musicians had started to experiment with simple 'sine-tone generators'. These were electric devices that would cause a metal strip to vibrate, thus producing a steady tone which could vary in pitch. This sound was often used in the musical scores of 1950s science fiction thrillers to suggest an eerie or futuristic atmosphere. It is still to be heard coming from television speakers as a signal to viewers to switch off their sets when transmissions are over. The first Hammond organs marketed in the 1930s were electronic and used this type of sound.

But it was the boom in electronics during the Second World War, specifically the German development of the tape recorder, which enabled musicians to create and manipulate sound in quite a different way. This could be done by splicing up analogue recording tape on which sound, 'musical' or otherwise, had already been recorded. These minute snippets of tape were then painstakingly combined to produce a collage of sound events. This 'new music' broke every rule

MARCUS WILSON-SMITH

Musica Obscura
New music demands a new notation. Stockhausen's scores with their pictorial representations of sounds, and graphic timing/synchronisation directions have nothing in common with classical scores, and were, indeed, intended to resemble electrical circuit diagrams

Ron Who?

The 'White Christmas' of electronic music must unquestionably be the 'Doctor Who' theme, written for the BBC Radiophonic Workshop in 1962 by Ron Grainer (seen here left of picture enjoying a joke with some of his chums on the set of BBC Television's 'Maigret' series)

© BBC TELEVISION

in the book as far as conventional music theory was concerned. It fascinated some listeners, and it tortured others.

At the same time, devices for varying and distorting the originally simple oscillator tones, and for filtering and modulating the result, were becoming more controllable and widely available. During the 1950s, composers such as Stockhausen in Germany were busy working in small studios attached to local radio stations, producing 'pure' electronic music. In Paris, working closely with sound engineers from ORTF, the French broadcasting company, Pierre Schaeffer pioneered what he termed '*musique concrète*', collage music using everyday sound from the real world.

In America, Bell Telephone Laboratories built what was probably the first synthesiser. It took up several rooms, and its primary purpose was to study human voice synthesis. The company knew that their telephone operators, from different parts of America, frequently misunderstood each other's accents, with a consequent high occurrence of false connections and wrong numbers. They hoped, perhaps a little optimistically for the time, that a universally accepted synthesised voice would clear up the problem. A number of today's American musicians received their basic training in electronics at that time, courtesy of the Bell company.

In Britain, similar work was being attempted, although on a less ambitious scale. Nevertheless, the BBC Radiophonic Workshop did produce one of the all-time classics of electronic music in the early 1960s — the theme music to the television series, 'Doctor Who'.

The first venture into computer music occurred as early as 1957, when Lejaren Hiller entered a set of instructions into the Illiac computer at the University of Illinois. These instructions were resolved into four groupings of technical data, which were then transcribed into musical notation. The result was a four movement work for string quartet called the 'Illiac Suite'. The music itself, though well-arranged for performance by cello, viola and two violins, sounds meandering and vague. However, it is not difficult to find other pieces of music, produced conventionally by composers of the same period, which sound a good deal worse.

A few years later, Hiller created another work, this time using the IBM 7090 computer. He designed a programming scheme called MUSICOMP (MUsic SImulator-Interpreter for COMpositional Procedures), which allowed for greater flexibility and variety in working towards the final composition. This he called 'Computer Cantata', and it is written for a vocalist performing with taped electronic sounds. Once again, the music is intermittently interesting rather than enthralling. But Hiller had demonstrated to his fellow musicians that a computer could be effectively used in a creative way.

His work was only part of a vast amount of research carried out in American universities in the ensuing years. John Chowning, another pioneer, later used a computer to explore how sound is perceived as its source moves from one location to another. Yamaha's use of his research work has had a direct bearing on the type of synthesiser being produced in the mid-1980s.

With the exception of music for science fiction features, electronic music stayed in the realm of classical music for several years, and audiences became more aware of this change in approach and technique on the part of avant-garde composers. A typical new music concert in the 1960s would feature several performers, some of them playing conventional instruments, others involved in processing the sound from those instruments with frequency-splitting units and filters. All of the performers, including the 'technicians', would be following a score, but this score bore little resemblance to standard music notation.

In addition to novel directions like those describing microphone positions and filter variations, composers were attempting to give

Pioneering Spirit

Probably best known for his work with Roxy Music in the early seventies, Brian Eno was a pioneer in the use of early synthesisers. After leaving the band in 1973, Eno has been a major force in avant-garde and 'mood' electronic music. He has also collaborated with such well-known figures as David Bowie and Robert Fripp. Most recently, Eno has worked on television and film scores, and with his brother has been developing a score for NASA's moon landing archive film

visual indications of what these new sounds were like in performance. In some cases, musicians were playing from music sheets that looked rather like a graphic designer's doodle-sheet. This problem — how to instruct music performance, what language to use, and how to visualise accurately the result — is one that still exists with digital music systems in the 1980s.

As the 1960s progressed, the pop musicians of the burgeoning youth culture began to spend more time in recording studios, and also started to experiment with electronic music. The classic example is that of the Beatles, who found in George Martin not only an expert recording engineer, but a musician who had kept a professional ear to developments in the classical field. He encouraged the Beatles to use the whole studio as a musical instrument, and before long they were using tape collage techniques and incorporating synthesised sound.

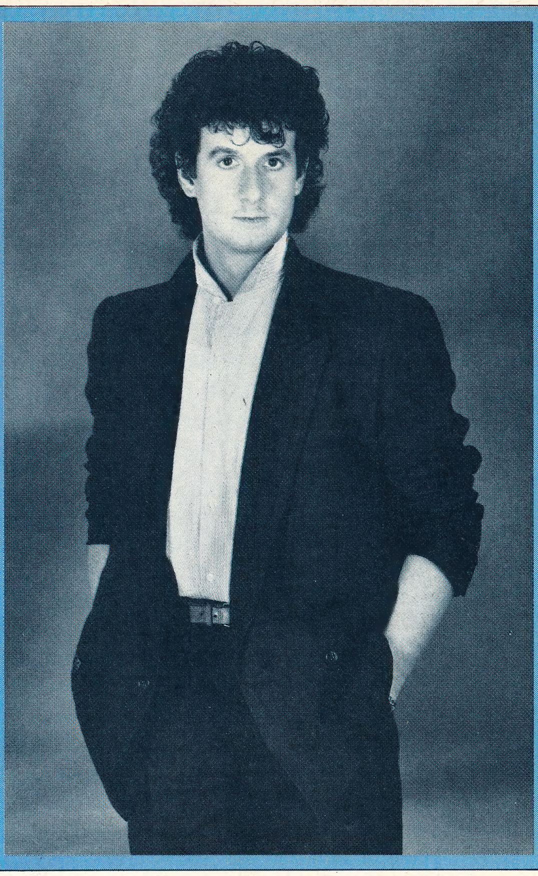
Some musicians made their names by using particular sound processing units. A guitarist like Jimmy Page modelled his playing style on that of black American musicians of the 1940s, but by using a series of distortion controls, produced a sound immediately identifiable as that of Led Zeppelin. This, coupled with Jimi Hendrix's use of feedback howl and rapid sweep filters (by now known as 'wah-wah pedals'), established Heavy Metal music.

By the 1970s, the various sound-generating and sound-processing units, which had been available since the 1950s, were incorporating transistor design. They became smaller, more portable, and as a result, less confined to a studio. Guitarists could play live using an assortment of effects pedals. Soon afterwards, organists and piano players had access to affordable synthesisers they could take on stage.

Typically, these synthesisers would include a set of tunable oscillators, envelope shapers (to create the attack, sustain and decay characteristics of the sound), variable filters, ring modulators (which could split signals up into new frequencies), and noise generators. Just as Jimi Hendrix had been a model for guitarists, Brian Eno became one for synthesiser players, chiefly because of his links with the 'new music' of the classical avant-garde.

At the same time, recording studio equipment became more sophisticated, as musicians looked to the studio to provide something in the production process that they could not create on stage. The mixing desk, now designed to channel successive recordings onto 16 or 24 tracks of tape, was still too large to carry around, and many of the processing units required time to set up. In America and Britain, a new breed of producers emerged. They had often started out as engineers, and had a deeper familiarity with the equipment than the musicians who paid them to provide 'the right sound'.

In Jamaica, engineers started to use the mixing desk as an instrument in itself. Completed songs, recorded on multi-track tape, would be stripped back to their individual rhythm tracks. The original vocal or instrumental contributions would then be used as raw material to drop in or out of the mix in a style heavily dependent on reverberation and signal delay units; this was the style known as 'dub'.



Ace Producer

Producer of the British band Culture Club, Steve Levine is most renowned for his ability to combine electronic music and human voices to produce 'seamless' (well-meshed) pop. Levine was one of the first producers in the UK to make use of the Linn drum, a programmable drum synthesiser, and has recently developed digital recording techniques to a fine art. Levine has made extensive use of synthesisers and other digital music equipment in his recent single, 'Believin'', co-written with Boy George

The advent of digital synthesisers brought the possibility of encoding non-electronic sounds. This process is known as 'sampling'. Drum machines like the Linn became sought-after studio items and very soon became part of stage performance. In the mid-1980s, sampling and manipulation of sound has become the 'state of the art', and well-equipped studios and stage set-ups generally contain more items of digital equipment than analogue instruments. Successful groups like Culture Club combine their own musical skills in songwriting with the digital production techniques of producers like Steve Levine, who uses instruments and processing units worth tens of thousands of pounds.

The necessity for an interface that could link up one instrument to another, or which could expand the capabilities of a synthesiser by adding the operating system and memory of a microcomputer, brought musical instrument manufacturers together. They came up with the first MIDI specification in April 1983 and, since then, few companies have dared to announce a new synthesiser that is not MIDI-compatible. In the next instalment in this series, we will look in more detail at the background and development of the MIDI.

FAULT LINES

The detection and correction of errors is an important aspect of program design. Problems may be caused by typing errors, but faulty logic or a misconception of the program's function can have more serious results. We examine potential trouble-spots — at the interfaces between subroutines and between the program and its user.

There are many potential sources of error at each stage of a program's creation, from its specification, through design and coding, to the testing. Errors are often introduced at the specification and design stages if little thought is given to the nature of the problem and insufficient care is taken to ensure that the program does exactly what it is supposed to do. We can reduce the chances of these mistakes occurring by following the structured design methods outlined earlier in the course (see page 476). Further errors are likely to arise as the design is translated into code — poor typing can introduce bugs, as anyone who has ever misspelt a variable name knows only too well! — and even testing and debugging can cause other mistakes when a correction to one fault itself leads to others.

But it is at the interfaces — between routines and between the program and its user — that most errors are to be found. Particular care should be taken to ensure that any values passed across these interfaces are of the correct data type and fall within the range required by the program. Values may be checked either within the routine that passes them or in the routine that accepts them; the process of checking values as they pass between routines is known as 'firewalling'.

To ensure that values output by a routine are in an appropriate range and are of the right data type, checks should be carried out if the output depends on a value entered by a user or read from a file. Values that are entered into a routine should always be checked. Subroutines can be designed to give a well-defined set of outputs, but human beings do not operate so methodically and tend to have a wide range of different responses to any given prompt, so stringent checks must be placed in any routines that accept data from users. Similarly, files of data may be corrupted or misread, so checks should be placed in all file-handling routines.

Errors do not often cause programs to crash. When they do, it is because the program has broken a rule of the language (using an operator illegally, for example, as in RESULT = FIRST$ + SECOND$) or a rule of the operating system (opening too many files at the same time, say). The following code would appear to be a perfectly legitimate program:

```
10 FORCOUNTER = 1 TO 10
20     SUM = SUM +1
30     PRINT COUNTER, SUM
40     GOTO 10
50 NEXT COUNTER
```

However, it is a non-terminating algorithm and will crash the system because of the way the language works. In this case, the language (BASIC) uses the 'stack' to keep track of FOR... NEXT loops, adding to the stack each time a new loop is started. In this program, line 50 (with the NEXT command that would decrement the stack) is never reached, and so the stack gradually fills up until eventually a 'stack overflow' message is generated and the interpreter stops the program. Errors such as this are usually easily spotted, but if they appear in rarely used sections of code thorough testing may be needed to uncover them.

A more insidious type of error is one that allows

Error Checklist

A logical structured approach is the essence of error avoidance and debugging; the following error checklist (from an idea by G J Myers in 'The Art Of Software Testing') is an abbreviated example of such an approach

Variables

1 Are all variable names unique, bearing in mind that many interpreters use only the first two characters of any name?
2 Have any variables (especially loop counters or subroutine parameters) been re-used while their contents are still significant?
3 Are array subscripts within bounds, and are they whole numbers?
4 Do array subscripts start at element zero or element one?

Calculations

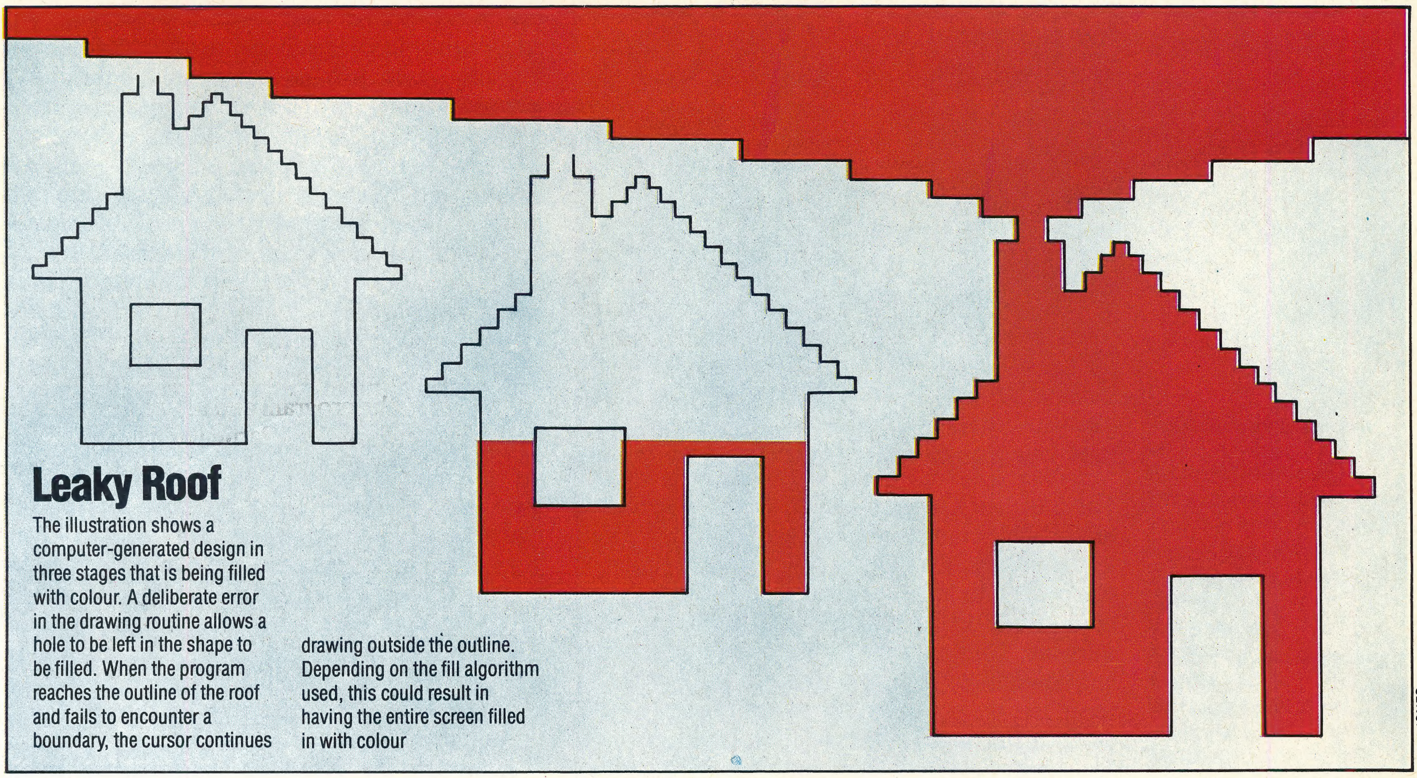
1 Do calculations yield string or numeric results, and are the results assigned to string or numeric variables?
2 Does any calculation result in a number too small or too large for the computer to handle? Can this cause a 'divide by zero' error?
3 Can rounding errors be significant?
4 Are all operations in an expression executed in the correct logical order, as opposed to the order imposed by the precedence of arithmetic operators?

Comparisons

1 Are strings always compared only with strings, and numbers with numbers?
2 Does it matter if a test string is wholly or partly upper-or lower-case?
3 Are strings of unequal length being compared, and does the difference in length matter more or less than differences in characters?
4 Are Boolean and comparison operators being mixed properly? A>B OR C is not the same as A>B OR A>C, for example.
5 Does the precedence of Boolean and comparison operators affect the execution of any comparison expression?

Control

1 Do loops and algorithms terminate whatever the state of the variables?
2 Do loops and routines have only one entry and exit point each?
3 When an IF...THEN statement fails, does control pass to the next program statement or the next program line?
4 What happens if none of the test conditions in a multiple branch statement is satisfied?



Leaky Roof

The illustration shows a computer-generated design in three stages that is being filled with colour. A deliberate error in the drawing routine allows a hole to be left in the shape to be filled. When the program reaches the outline of the roof and fails to encounter a boundary, the cursor continues drawing outside the outline. Depending on the fill algorithm used, this could result in having the entire screen filled in with colour

KEVIN JONES

a program to run normally, but invalidates the results. As an example, we have chosen to look at a fill pattern that draws a shape on the screen, then fills it with colour. Fill routines look for the boundaries of the shape. When a boundary is reached, the computer turns the cursor around and continues drawing until it reaches another boundary. For a fill routine to work, the boundaries must be well-defined and complete. In other words, there cannot be an open space in the shape's outline or the fill routine will spill the colour out beyond the boundaries.

The versions of the BASIC language used by most home micros make error-handling relatively easy, producing clear and concise error messages and allowing a crashed program to be continued after variable values have been altered at the keyboard — a useful facility when a program is being debugged. Most BASIC dialects will allow the use of a command such as ON ERROR GOTO to transfer the flow of control to a special error-handling routine and thus deal with otherwise 'fatal' bugs. This is done by including a program line such as:

```
30 ON ERROR GOTO 20000: REM error-handling
   routines
```

near the start of the program. Any error will then cause the program to act as though the GOTO 20000 command had been encountered. ON ERROR will usually also set two variables; the first of these stores an error number that indicates the type of error that has occurred, and the other simply holds the line number at which the mistake was encountered. The names given to these variables and the resulting error numbers will vary from machine to machine, so the manual must be consulted. Once an error has occurred, program flow is diverted to line 20000, the error is identified from the number held in the relevant variable and the appropriate action is taken.

A well-written program will not have more than one ON ERROR routine. Such a routine will not be able to deal with syntax errors, memory shortages, stack overflows, etc. The best that this facility can offer is an orderly shutdown of the system, ensuring that all files are CLOSEd and that the user knows exactly what has happened.

Some errors, such as a division by zero, which could be handled by such a routine, should in fact be dealt with in a different manner. There are several reasons for this:

- The ON ERROR GOTO command and the subsequent jump back to the main program constitute an extra entry to and exit from a routine. This violates the structured programming principle that routines should have only one entry and one exit point.
- The proper place to protect against a division by zero is in the routine that does the division. It is bad practice to design algorithms that may crash the system. If the extra error-checking involved slows the program to an unacceptable degree, the routine should be redesigned so that this hazard doesn't arise.
- Error-handling routines rapidly become complicated IF...THEN...ELSE chains with multiple exits. They are inevitably restricted by the line numbering of the rest of the program and so must be rewritten whenever any routine using them is redesigned. They are particularly difficult to design, test and debug, and any mistake in such a routine can introduce far-reaching problems by diverting the flow of control in unforeseen ways.

BREAKING AND ENTERING

Gaining illicit access to mainframe machines using home computers and modems is known as 'hacking'. In recent years there have been a number of celebrated cases involving government departments and multinational corporations, and hacking has now become a topic of public concern.

The film *War Games* captured the imagination of many home computer owners. Using a micro and a modem, the hero illicitly dials up a succession of computers to change his college exam results, book airline tickets and download the latest games software. Things start to go wrong, however, when he unwittingly gains access to the NORAD computer responsible for North American air defence, and almost starts a global nuclear war. A fine piece of entertainment, but surely it's much too far-fetched?

Illegal Distribution

Athough the Pepsi-Cola Company of North America deny any knowledge of the incident, they were the victims of a celebrated case of hacking in recent years. Reports claim that someone in the United States illegally accessed a Pepsi-Cola company computer in Canada. The hackers sent large shipments of Pepsi to assorted destinations as a means of moving large sums of money into illicit accounts

IAN McKINNELL

Many such computer 'break-ins' have actually happened, with the culprit often turning out to be a teenager using a home micro and modem. The 'victims' range from powerful mainframe computers belonging to universities and large corporations to the bulletin board services run by enthusiasts on microcomputers. Any computer that allows external access by telephone is vulnerable. In 1983, reality came close to imitating fiction when it was suspected that two 'hackers' had succeeded in accessing the NORAD computer system in Omaha, Nebraska.

The two teenagers involved came from Los Angeles and had managed to get into Arpanet — the secret computer network run by the Defense Department in the United States. Using a Commodore Vic-20 and a Tandy TRS-80, the pair managed to explore the contents of several of the computers connected to Arpanet, which typically belong to defence contractors, research organisations and universities. Although no classified information was obtained — the system is used mainly for sharing scientific data — the ease with which the two teenagers 'broke in' caused major embarrassment to the Defense Department.

The reason the boys found it so easy had more to do with human laziness than any computer fault. Registered Arpanet users all have passwords; unfortunately, these were not chosen very imaginatively. In this case, the two boys guessed that the University of California at Berkeley might be an Arpanet user. Sure enough, the password 'UCB' got them into the network and then they were free to access any of the computers connected to Arpanet — one of which is the NORAD underground headquarters in Omaha.

Although the NORAD headquarters is on Arpanet, the computers responsible for actual air defence are not. They sit under the Cheyenne Mountains in Colorado and are not connected to the public telephone lines.

NORAD computers might be immune to violation from hackers, but many others are not. In another incident, in July 1983, a group of Milwaukee teenagers broke into more than 60 computers belonging to colleges, corporations and the Los Alamos National Laboratory, which is engaged in weapons production. Again, according to the authorities, no classified information was obtained — just records, routine reports and messages. The FBI was called in to find out how the group, who called themselves the '414s', had pulled off such a feat. The 414s said that there were no security measures on any of the computers they phoned up.

These are just some of the cases that have been publicised. Many others are not made public, as few organisations want it known that their mainframe computer has been infiltrated by, say, a 17-year-old with a £100 home micro. Also, many organisations simply are not aware of infiltration: it is often very difficult to know if an unregistered user or impostor has been 'on-line' — though some of the more cheeky hackers leave 'can't catch me' messages and sign-off as 'System Crasher' or 'Captain Zap'.

How exactly is hacking done? All the potential hacker needs is a home computer, a modem and a bit of ingenuity. The first hurdle is finding a computer's telephone number. For public-access networks, such as Telecom Gold in Britain or The Source in the USA, this is not difficult as they are usually widely published. For private computers it's more difficult. But if you know roughly where the computer is situated, then the technique used by the protagonist in *War Games* could be used: he programmed his micro to dial every possible telephone number in his town. If a computer answered — identified by the tell-tale carrier tone whistle — then the machine took a note of the number; but if a person answered, the modem hung up and went on to the next number. With an auto-dial modem this can be done automatically; dialling manually would get very tedious!

Once connected with the computer, you are invariably asked for a password. Some networks allow limited access if you type in 'GUEST' or 'NEWUSER', or if you just press RETURN. But the true hacker will try and crack the password. Often this is not particularly difficult, as users tend to be rather unimaginative and use names, such as 'SMITH', or obvious words, like 'SECRET', or even, simply, 'PASSWORD'. Similarly, with passwords made up from numbers, people tend to choose easy-to-remember sequences: for example, their date of birth, like '090560'. Many computers are very forgiving and will allow several attempts at the password before disconnecting you. Even then, you can normally dial back and continue where you left off, without the host computer becoming suspicious.

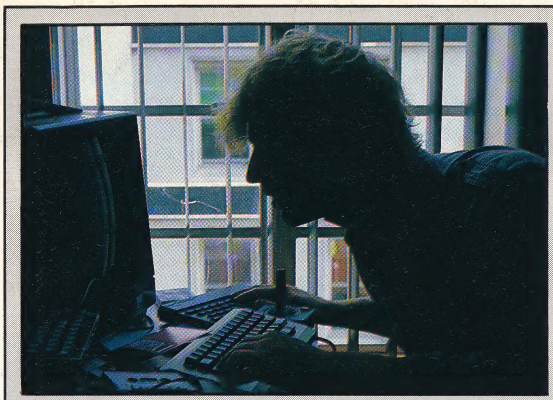
Once into a system, most hackers are content to just look at everyone's files, find the games pages (if any) and 'talk' to other hackers who have also broken in. Some of the more destructive ones delete files, leave obscene messages and try to 'crash' the whole system. A system crash can have a disastrous effect on legitimate users.

Even on the more sophisticated mainframe machines, programmers often leave 'back doors' in the system so that in an emergency they can bypass all the protection measures and quickly get into the program. More often than not, the people who operate the system will not know that such back doors exist.

You will notice that most of the cases we have outlined involve university computers. This is because such computers, apart from having external dial-up access, usually operate an open access policy. With thousands of users and many remote sites this is the most practical way to run such a system. Unfortunately, they are also very easy for hackers to get into, and once in they can 'leap-frog' from one computer to another by posing as legitimate users. One student at San Jose State campus found a loop-hole in the university computer's Talk program, which allows students to 'talk' to the other campuses in the California State University. The student managed to overcome the local restriction and succeeded in talking to computers in Sweden, Iran and China, as well as all over the United States. The telephone bill came to over £7,000.

Why do hackers do it? Usually just for the thrill of 'beating the system'. Many have an unofficial code of conduct, and claim not to delete files or leave obscene messages. For them, the excitement is simply in breaking the codes. Nevertheless, by

Trial And Error



IAN McKINNELL

The process a hacker uses involves a great deal of trial and error, which on very rare occasions leads to successful unauthorised entry into a system. Even if a hacker is able to locate a particular system, he is often met with a dialogue like this when he tries to log on to the system:

```
CONNECT QX001001 14.32
      12/7/84
>
USER?
>HELP
USER?
>£$OFF
USER?
>UK001
PASSWORD?
>GUEST
PASSWORD?
>NEWUSER
PASSWORD?
>QWERTY
LOGOFF 15.13 CONNECT
      TIME = 0.13 MINS
```

- Press return: prompts for user ID
- No help: not user friendly
- First attempt at a valid ID
- ID not valid for entry
- Second attempt at a valid ID
- ID accepted: prompts for password
- First guess
- Not accepted: second request
- Second guess
- Not accepted: third request
- Desperation guess
- The user is disconnected after the third failed attempt to find a password

In some cases, whether by intrigue or pure luck, a person is able to find a valid high-priority password and enter a system. The following imaginary dialogue describes how such a user could learn confidential information about a legal user:

```
CONNECT BYF990
      15.14.02 12/07/84
>
USER?
>UK001
PASSWORD?
>SYSOP
LOGON 15.15.07 12/07/84
HOST: BYF990/SPYLOM
USER: UK001
SERIAL NO: ZA180-7
PRIORITY: SUPERUSER
STATUS: ACTIVE
YOU ARE SYSOP
7 USER(S)
APP01 APP02 BYF7 BTY04
      BZX88 BZX02 SYSOP
>REMOVE ARP01
USERS(S) ARP01
      DISCONNECTED
>WHO IS BTY04
BTY04     CAREY DIMMIT,
      BTY LOPP, 742
      SILICON DV
HERTS 07662-093164
```

- Press return
- Valid ID: prompts for password
- First attempt: system operator
- Serial number of software
- Full clearance to files
- Users can be made inactive if they don't use the system regularly
- Confirmation
- Lists all users
- Command reserved for system operator
- Valid user removed

using computer time and not paying for it they are committing fraud.

Banks have long suffered from computer crime, but until recently it was all done 'in house' — by dishonest employees transferring money to bogus accounts, for example. Estimates for computer fraud vary from £30 million to over £2,500 million per year — and that's in Britain alone. Understandably, banks and companies seldom publicly acknowledge that they have been victims of computer fraud and hence it is difficult to put an exact figure on it.

With the growth in ownership of home computers, and with more and more computer networks using the telephone lines, the problems of computer crime can only increase. Whether it be mischievous teenage hackers deleting files and stealing computer time or professional criminals siphoning money into their own accounts, the methods used are exactly the same.

D

DROP-IN

A *drop-in* is a piece of unexpected data that appears on a magnetic recording medium, such as a floppy disk or cassette tape. Its signal is picked up by the system even though the data was not intended to be written there. The presence of a drop-in is usually the result of a fault in the recording surface causing the incomplete erasure of data that had previously been recorded there. As with drop-outs, the existence of a drop-in does not usually affect the correct flow of data because most operating systems can identify stray bits and check the information.

One way to prevent the occurrence of drop-ins on cassette tape is to erase carefully the surface before recording new data. It is common simply to record over existing information, but doing so increases the risk of incomplete erasing.

DROP-OUT

A *drop-out* occurs when a piece of magnetic material has flaked off the recording medium — cassette or disk — used to store programs. Cheap cassettes are particularly prone to this, and such drop-outs can render a recorded program useless. This is why magnetic media should always be handled with care.

Floppy disks and cassettes that develop drop-outs are best discarded, but what happens if they occur on a winchester disk, a unit costing a couple of thousand pounds? The answer is that winchester drives have a more sophisticated form of DOS than the less expensive microcomputer disk drives. If a bad block is encountered (i.e. the drive is reading back incomplete data), the DOS keeps a note of its location and doesn't record anything there again.

DUMP

A *dump* is the technical term for a visual listing of the contents of an area of memory. This may be displayed on the screen or output to a printer. A 'binary dump' lists the contents of each specified byte in binary, a 'hex dump' in hexadecimal, and so on. Apart from listing machine code programs in a form that can be easily entered into a monitor program, dumps are very useful for debugging. They allow the programmer to see what effect the program has had on the data held in memory.

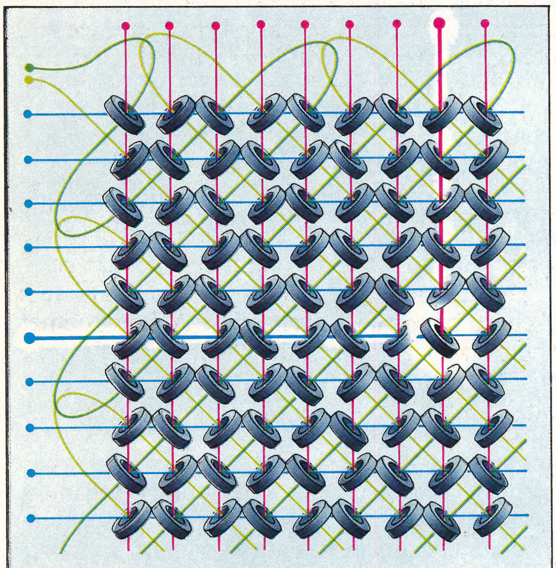
DUPLEX

A communications channel is *duplex* if data can be transmitted in both directions simultaneously. Half-duplex means that data may travel in either direction but not in both at the same time. Many walkie-talkie radios and intercoms work in half-duplex mode, and require that the two parties take turns in transmitting and receiving (hence the need for saying 'over' to mark the end of transmission). Early forms of computer communications over telephone lines were half-duplex only, and some modems still feature a switch to enable duplex or half-duplex options to be selected. Full duplex requires a greater bandwidth (see page 148) for transmission.

It is rare for useful information to be transmitted in both directions at once, but simultaneous data transmission allows the receiving computer to send messages about the incoming signal back to the transmitter. If noise on the line has produced errors in the signal, the receiving device can request that a particular block of, say, 128 characters be sent again.

DYNAMIC RAM

Two types of RAM are commonly found in microcomputer systems — static and *dynamic*. Both are volatile, which means that they will lose their contents if power is disconnected. Static RAM is constructed from bistable circuits (see page 168) — one bistable, or flip-flop, is used for each bit of memory. A static RAM chip requires virtually no additional electronics outside the chip to interface it to the microprocessor.



Core System
The acronym RAM nowadays always refers to electronic microchip devices, but for 20 years the ferro-magnetic core system illustrated here was the cheapest, fastest, most elegant solution to the random access memory problem. Binary data was represented by the magnetic polarity of the iron rings

Dynamic RAMs are more complex in operation, but they are faster, cheaper and allow more bits to be fitted onto one chip. Each bit is really a small capacitor, which is electrically charged to represent a binary one or left uncharged to indicate a binary zero. This charge is not permanent, and special 'refresh' circuitry is used to 'top up' the RAM at intervals of a few milliseconds. Newer designs incorporate the refresh circuitry inside the chip itself, making dynamic RAMs easier to integrate into systems.

If the top is removed from a transistor, the current passing from the emitter to the collector will be affected by light falling on the device. Some early types of dynamic RAM are similarly light-sensitive, and may be used as the basis of low-cost robotic vision systems.

RELATIVE NEWCOMER

A recently-released British-made microcomputer breaks new ground by being the first home machine to feature a built-in disk drive. The Tatung Einstein offers 80 Kbytes of RAM and has a full range of interfaces for expansion. It also has a comprehensive BASIC, good graphics and sound.

The Einstein's price tag means that it will appeal mainly to the 'serious' home user. It may certainly be used for playing games, but it offers few advantages in this sphere over machines costing a quarter of the price. Its closest rival, in price and performance, is the BBC Micro, but the Einstein's 80 Kbytes of RAM compares favourably with the somewhat meagre 32 Kbytes offered by Acorn.

The disk drive is mounted in a panel just above the keyboard in the Einstein's unusually large casing. This casing is strong enough to take the weight of a monitor or television, so the complete system does not take up too much desk space.

The major advantage of the integral disk drive is likely to be software availability. Although other home machines, such as the Commodore 64, may be fitted with disks, the fact that they are designed with cassette recorders in mind means that most software is produced on cassette and disk owners will therefore be unable to make the most of their superior storage medium. The Einstein's built-in drive ensures that all software will be supplied on disk from the start. The use of disks allows programs and data to be loaded quickly and reliably and enables random access files to be used instead of the serial access files to which cassette-based machines are restricted. 190 Kbytes of data may be stored on each side of the 3in disk, but the Einstein can use only one side at a time. A second drive may be fitted into the casing at a cost of £150, and two further drives (£190 each) can be plugged into an interface at the rear of the machine.

To control disk use, the Einstein has its own disk operating system (DOS). This has many similarities to the CP/M standard used by many business machines, and Tatung hopes that software houses will convert CP/M programs to run on the Einstein. The operating system and Einstein BASIC are not held in ROM, as is usually the case, but must be loaded from disk each time the machine is used. This has two main advantages: as BASIC is loaded only when it is needed, other programming languages or machine code programs can utilise the full RAM space, and both operating system and BASIC can easily be upgraded by simply purchasing a disk containing the new versions. Tatung plans to offer the DR LOGO language on the free disk supplied with the machine — but this is not quite as generous as it sounds because the LOGO manual will be sold separately for around £25. Once BASIC has been loaded from disk, the Einstein has a healthy 43 Kbytes of RAM available to the user, which is more than is offered by any other home machine. This is possible because the Einstein RAM contains a separate 16 Kbytes of memory that is controlled by the graphics chip and which is used to handle the screen display.

Einstein BASIC appears to be a blend of BBC BASIC and Microsoft Extended BASIC (as used by the Japanese MSX machines). It includes commands to renumber programs and to produce line numbers automatically, thus making the keying-in of programs easier. A full screen editor allows changes to be made anywhere on the screen display. The graphics commands permit lines, circles and ellipses to be drawn and enable outline shapes to be filled with solid colour. Graphics have a maximum resolution of 256 × 192 pixels and

Serious System
The Einstein computer from Tatung (formerly Decca). The machine is aimed at the serious home user, and comes equipped with a disk drive. It is larger than most home machines, enabling the monitor to rest on the casing

CHRIS STEVENS

there are 15 colours available, although no more than two of these may be used in each row of eight pixels. This limitation is imposed by the graphics chip used. Up to 32 sprite characters may be defined by the user; the commands to control these are included in the BASIC, which makes it possible to write impressive games programs with fast-moving action. A monitor program is provided in ROM to make machine code programming easier.

The graphics chip also limits the display to a maximum of 40 characters in width. A £50 add-on is planned to allow the 80-character display required by many CP/M programs to be duplicated on the Einstein; the 80-column screen will be monochrome only, but a colour version is planned for 1985.

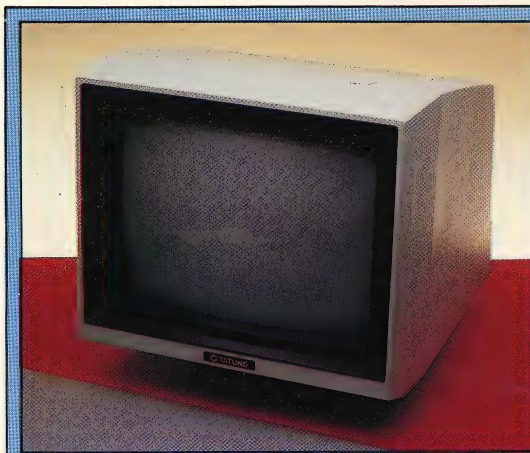
The Einstein's sound quality is good, with output directed to a large loudspeaker located above the keyboard. A volume control is provided, and the BASIC commands to generate sound are comprehensive and easy to use.

Eight function keys are provided; these may be programmed to produce commonly used words and commands and a clear plastic strip allows labels to be fixed above each key, BBC fashion. The keyboard is well constructed and touch-typing should be no problem, but Tatung has provided only two cursor keys instead of the more usual four. This means that cursor keys must be used in conjunction with the shift key to produce movement in two of the four directions, which is extremely irritating on a machine of this price. A set of graphics characters can be produced from the keyboard if the graphics key is held down, but these are of limited use.

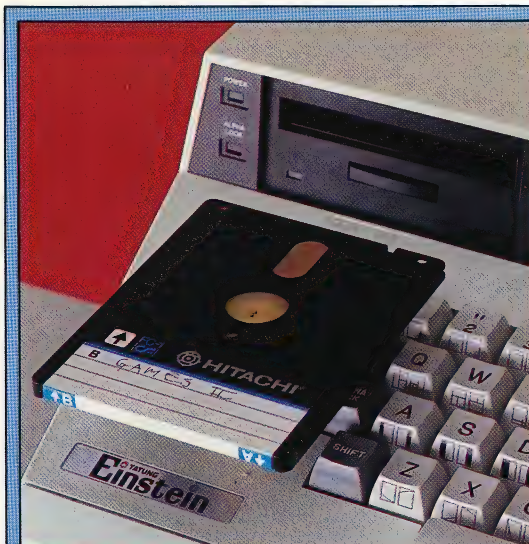
The Einstein is rivalled only by the BBC Micro in the number of interfaces offered. These include a standard Centronics interface for printer connection; an RS232 socket for use with printers, modems and other accessories; a socket for an RGB colour monitor (Tatung supplies one for £240); an output for a television display; and a pair of joystick sockets. The joystick sockets may also be put to more serious use as they are of the analogue-to-digital converter type that allows electrical voltages to be measured. The two sockets provide four A-to-D channels; these are complemented by an eight-bit user port that can input and output digital signals to and from other items of electrical equipment. This combination of user and A-to-D ports makes the Einstein ideal for control uses in robotics and scientific applications.

Future expansion is made possible by the 'Pipe' (similar in concept to the 'Tube' of the BBC Micro), which will allow various add-ons to be fitted. A ROM socket inside the machine allows the eight Kbytes of ROM fitted as standard to be expanded to 32 Kbytes.

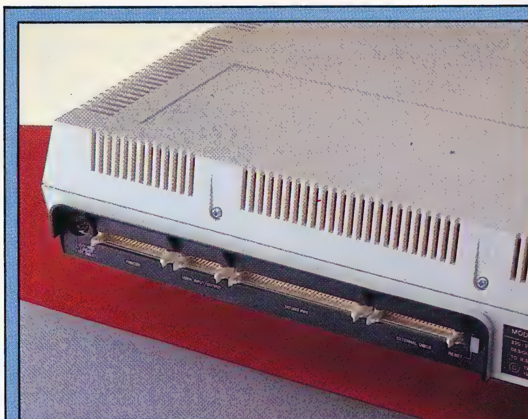
The British-made Einstein is undoubtedly good value for money, but the fact remains that few users can afford to spend £500 on a home computer. Little software is as yet available, and this situation is unlikely to be rectified unless the machine sells in large numbers.



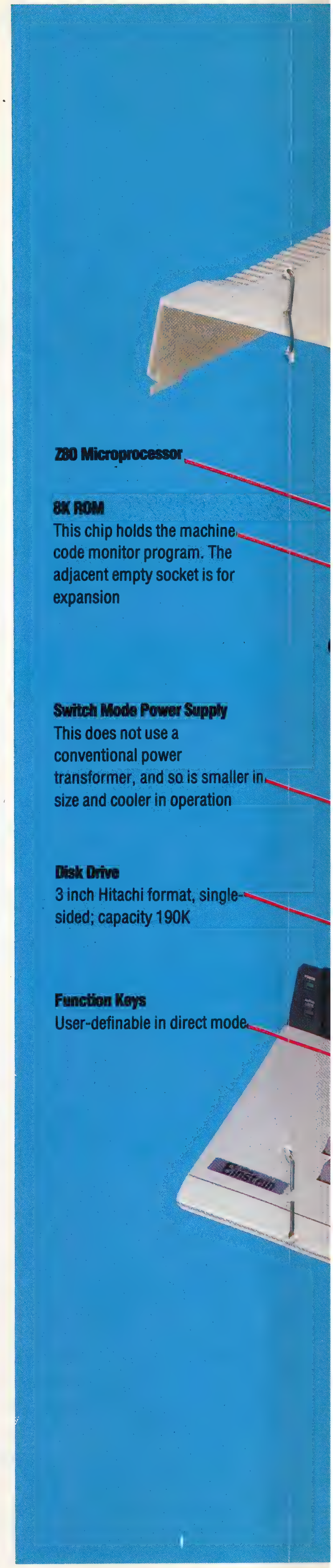
Monitor Options
The Tatung monitor accepts RGB or YUV input; the latter is Tatung's own, supposedly superior, system. Since YUV output includes a composite video line, any monitor can be used with it

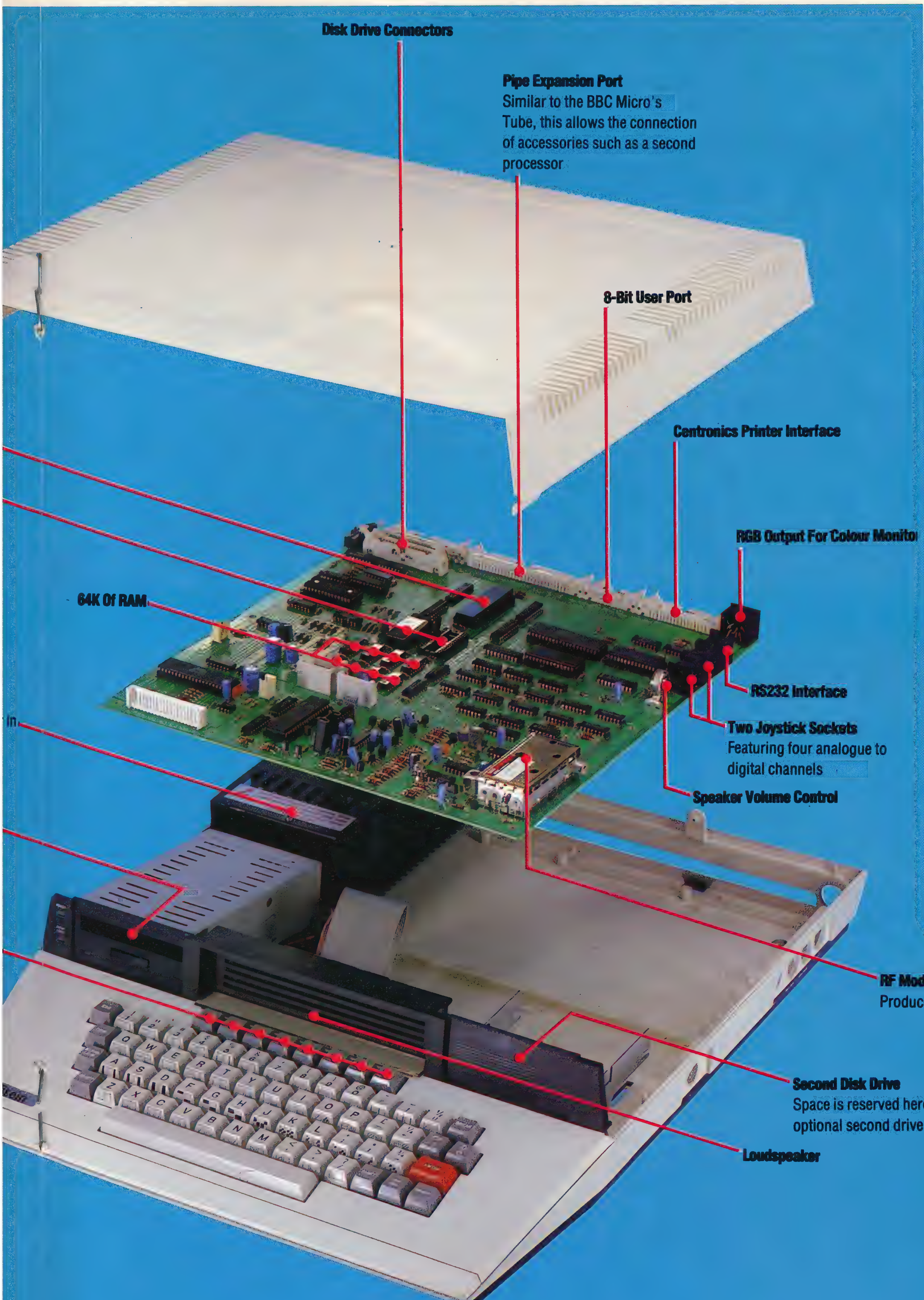


Hitachi Drive
The disk drive is the Hitachi 3in system, which is becoming increasingly popular on microcomputers. The disks can store up to 190 Kbytes, and although the drive reads only one side, the disks can be manually turned over so that both sides can be used



Einstein Interface
The Einstein is well equipped with interfaces including the Tatung 'Pipe', a general-purpose port similar to the BBC Micro's 'Tube'. There is also an interface for additional disk drives





TATUNG EINSTEIN

£500

PRICE

510 x 430 x 105mm

DIMENSIONS

Z80

MEMORY

80K of RAM, including 16K screen memory. 8K ROM, expandable to 32K

SCREEN

Text: 32 columns x 24 rows/40 columns x 24 rows. Graphics: 256 x 192 pixels with 15 colours

INTERFACES

Centronics, RS232, two joysticks (A-to-D), RGB, 'Pipe', user port and external disk

LANGUAGES AVAILABLE

BASIC and DR LOGO supplied. PASCAL and FORTH available

KEYBOARD

67 typewriter-style keys, QWERTY layout. Includes eight programmable function keys

DOCUMENTATION

Three manuals including poor BASIC tutorial

STRENGTHS

Built-in disk drive, many interfaces, large user memory, good BASIC, sprite graphics

WEAKNESSES

Little available software. Retail price higher than other home machines

CHRIS STEVENS

OUT WITH A BANG

We put the finishing touches to our Mines game for the BBC Micro and the Electron. In particular, we look at the BBC Micro's teletext mode, and use this in our 'end-of-game' screen display. Finally, we conclude the project with a full listing of the program, and give the necessary alternative lines for the game to run on the Electron.

The BBC Micro's graphics mode 7, also known as the teletext mode, has several features that are not present in any other mode. These features are provided for the display of transmitted information from external sources such as Micronet, which can be accessed using the computer and a standard telephone line. The additional graphics features that can be accessed using mode 7 can produce attractive lettered displays with a few simple instructions and, consequently, this mode is an ideal choice for our 'end-of-game' screen.

By using CHR$ control codes embedded in PRINT statements, we can control the text and background colours, create 'flashing' text and produce double-height characters. We can use the TAB function in the usual way to position text on the 40 by 25 character screen. Seven colours are available and can be selected by the following control codes:

129	red
130	green
131	yellow
132	blue
133	magenta
134	cyan
135	white

Whenever mode 7 is used, the text is displayed in white on a black background. The text colour can be changed at any point during a PRINT statement by inserting a control code. For example, the three words of the phrase in the following line will be printed in red, white and blue, respectively:

```
PRINT CHR$(129)"JEUX";CHR$(135);"SANS";
      CHR$(132);"FRONTIERES"
```

It is important, however, to realise that when another line is PRINTed, the default colour (white) will be restored. We must therefore use control codes on each new line, even if we wish to continue PRINTing in the same colour.

As well as being able to select text colour, we can also select background colours. CHR$(157), followed by the colour we want for the background, allows us to do this. For example, to produce blue letters on a white background, the following combination of control codes are used:

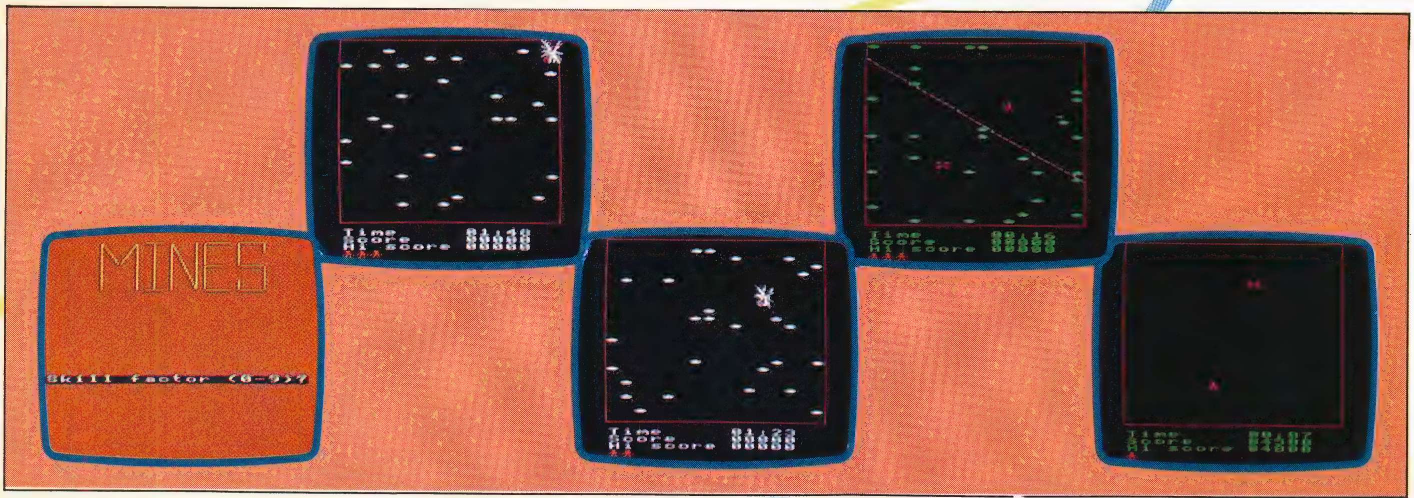
```
PRINT CHR$(132);CHR$(157);CHR$(135);"IN THE
      NAVY"
```

The first control code specifies the colour of the text, the second and third define the background. Both text and background can be made to flash by using a CHR$(136) control code immediately before the colour code. CHR$(137) turns this effect off again. For example, we can make the blue letters in the last example flash by using:

```
PRINT CHR$(136);CHR$(132);CHR$(157);
      CHR$(135);"IN THE NAVY"
```

The most impressive feature of mode 7 is its ability to produce double-height characters. CHR$(141) allows us to do this, but we must also PRINT the same line twice to achieve the correct effect. We can attribute all the other effects to double-height characters. For example, to produce double-

Mind Your Step
These frames from several runs of the game show the mines, the assistant, sniper fire, an explosion and the score and title displays



height, flashing, blue characters on a steady white background, we use the following lines of code:

```
10 MODE 7
20 PRINT CHR$(141);CHR$(136);CHR$(132);
   CHR$(157);CHR$(135); "IN THE NAVY"
30 PRINT CHR$(141);CHR$(136);CHR$(132);
   CHR$(157);CHR$(135); "IN THE NAVY"
```

You can see that all these control codes take up a lot of program space and are time-consuming to type out. An alternative method is to chain several codes together into a single string that can then be used in the PRINT statements. For example, if we need to do a lot of PRINTing using red characters on a yellow background, we can start by creating a string (red$) that can then be used in each PRINT statement requiring this effect. Thus:

```
10 MODE 7
20 red$=CHR$(129)+CHR$(157)+CHR$(131)
30 PRINTred$; "GAME OVER"
40 PRINTred$; "YOUR SCORE"
```

THE END-OF-GAME PROCEDURE

Let's now look more closely at how we employ these effects in our end-of-game screen:

```
2110DEF PROCend_game
2120 IF score$>hi_score$ THEN hi_score$=score$
2130red$=CHR$(129)+CHR$(157)+CHR$(131)
2140game$="G A M E  O V E R "
2150PRINTTAB(0,5)red$;CHR$(141);CHR$(136);TAB(12);game$
2160PRINTred$;CHR$(141);CHR$(136);TAB(12);game$
2170PRINT:PRINTred$;"Your Score";TAB(30);score$
2180PRINT:PRINTred$;"Hi score";TAB(30);hi_score$
2190PRINT:PRINTred$;"Time";TAB(30);time$
2200blue$=CHR$(132)+CHR$(157)+CHR$(134)
2210go$="A N O T H E R  G O    Y / N ?"
2220PRINT:PRINT
2230PRINTblue$;CHR$(141);CHR$(136);TAB(5);go$
2240PRINTblue$;CHR$(141);CHR$(136);TAB(5);go$
2250REM ** REPLY ? **
2260*FX 15,1
2270answer$=INKEY$(0)
2280IF GET$="N" THEN finish_flag=1
2290ENDPROC
```

Line 2120 checks to see if the score in the game just concluded was greater than the previous highest score and updates the high score if necessary.

The message GAME OVER is then printed in double-height, red flashing characters on a yellow background (lines 2130 to 2160) and the details of the scores and time are displayed (lines 2170 to 2190). The player is then asked if another game is required. If the answer is N then a variable (finish flag) is set to one.

Notice that mode 7 has not been set during this procedure. This is because the BBC Micro will not allow a mode change to be made within a procedure. An attempt to do this will result in a BAD MODE error message. We must, instead, set mode 7 in the short main program that calls the procedures. The following lines should be added to complete the program. Notice that the whole of the calling program has now been placed in a REPEAT . . . UNTIL loop, which will repeat until finish flag is set to one.

```
1100REPEAT

1200MODE7
1210REM ** TURN OFF CURSOR **
1220VDU23,1,0;0;0;0;
1230PROCend_game
1240UNTIL finish_flag=1
1250CLS
1260END
```

THE ELECTRON ALTERNATIVE

Electron users may have been a little worried during our discussion of mode 7, as the Electron does not feature this mode. As an alternative, we have prepared a different procedure that uses mode 5 for the end-of-game screen. Omit line 1200 from the calling program just given and enter this procedure in place of the BBC's end-of-game procedure:

```
>L.2100,2300
 2100 DEF PROCend_game
 2110 IF score$>hi_score$ THEN hi_score$=score$
 2120 REM ENSURE BACKGROUND YELLOW
 2130 VDU19,130,3,0,0,0
 2140 GCOL0,130:CLG:REM COLOUR SCREEN
 2150 COLOUR1:COLOUR130:REM SET TEXT COLOURS
 2160 game$="G A M E  O V E R "
 2170 PRINTTAB(2,4);game$
 2180 COLOUR0
 2190 PRINTTAB(0,8);"Your Score";TAB(15);score$
 2200 PRINT:PRINT"Hi score";TAB(15);hi_score$
 2210 PRINT:PRINT"Time";TAB(15);time$
 2220 go$="ANOTHER GO Y/N ?"
 2230 REM CHANGE COL3 TO FLASH YELL/BLUE
 2240 VDU19,3,11,0,0,0
 2250 COLOUR3
 2260 PRINT:PRINT
 2265 PRINTTAB(2)go$
 2270 REM ** REPLY ? **
 2275 *FX 15,1
 2280 answer$=INKEY$(0)
 2285 IF GET$="N" THEN finish_flag=1
 2290 VDU 20:REM RESET DEFAULT COLOURS
 2300 ENDPROC
```

The Final Listing

```
1000REM ********************
1010REM **              **
1020REM **    MINES     **
1030REM **              **
1040REM ********************
1050:
1060hi_score$="00000"
1070finish_flag=0
1080:
1090REM **** MAIN PROGRAM ****
1100REPEAT
1110MODE5
1120REM ** TURN OFF CURSOR **
1130VDU23;8202;0;0;0;
1140PROCtitle_page
1150CLS
1160PROCsetup
1170:
1180PROCloop
1190:
1200MODE7
1210REM ** TURN OFF CURSOR **
1220VDU23,1,0;0;0;0;
1230PROCend_game
1240UNTIL finish_flag=1
1250CLS
1260END
1270:
1280:
1290REM **** DEFINE PROCEDURES ****
1300DEF PROCtitle_page
1310GCOL 0,129
1320CLG
1330GCOL 3,3
1340PROCmusic
1350Y=100:X=0
1360REPEAT
1370X=X+20:Y=Y+50
1380FOR I=1 TO 2
1390PROCmines
1400NEXT I
1410UNTIL Y>700
1420:
1430PROCmines
1440PRINTTAB(0,20)"Skill factor (0-9)?"
1450PROCmusic
1460REPEAT
1470skill=GET-48
1480UNTIL skill>-1 AND skill<10
1490ENDPROC
1500:
1510DEF PROCmines
1520PLOT4,X,Y
1530REM ** LETTER M **
1540PLOT1,0,200
1550PLOT1,80,-100
1560PLOT1,80,100
1570PLOT1,0,-200
1580REM ** LETTER I **
1590PLOT0,40,0
1600PLOT1,80,0
1610PLOT0,-40,0
1620PLOT1,0,200
1630PLOT0,-40,0
1640PLOT1,80,0
1650REM ** LETTER N **
1660PLOT0,40,-200
1670PLOT1,0,200
1680PLOT1,120,-200
1690PLOT1,0,200
1700REM ** LETTER E **
1710PLOT0,160,0
1720PLOT1,-120,0
1730PLOT1,0,-200
```

Final Listing
This shows clearly the spelling of those variable and procedure names (such as hi_scores$ and PROCend_game), which include the underscore character, "_". BBC Micro programmers will be familiar with its use as a legal spacing character; it should not be confused with the hyphen

```
1740PLOT1,120,0
1750PLOT0,-40,100
1760PLOT1,-80,0
1770REM ** LETTER S **
1780PLOT0,280,60
1790PLOT1,0,40
1800PLOT1,-120,0
1810PLOT1,0,-100
1820PLOT1,120,0
1830PLOT1,0,-100
1840PLOT1,-120,0
1850PLOT1,0,40
1860ENDPROC
1870:
1880DEF PROCsetup
1890COLOUR 2
1900end_flag=0
1910PROCinitialise_variables
1920PROCdefine_characters
1930factor=skill*3+30
1940PROClay_mines(factor)
1950PROCdraw_border
1960PROCset_time
1970PROCset_score
1980PROCset_men
1990PROCposition_chars
2000ENDPROC
2010:
2020DEF PROCloop
2030REPEAT
2040PROCupdate_time
2050PROCtest_keyboard
2060rand=RND(50-skill)
2070IF rand=1 THEN PROCsnipe
2080 UNTIL TIME>12099 OR end_flag=1
2090ENDPROC
2100:
2110DEF PROCend_game
2120 IF score$>hi_score$ THEN hi_score$=score$
2130red$=CHR$(129)+CHR$(157)+CHR$(131)
2140game$="G A M E  O V E R "
2150PRINTTAB(0,5)red$;CHR$(141);CHR$(136);TAB(12);game$
2160PRINTred$;CHR$(141);CHR$(136);TAB(12);game$
2170PRINT:PRINTred$;"Your Score";TAB(30);score$
2180PRINT:PRINTred$;"Hi score";TAB(30);hi_score$
2190PRINT:PRINTred$;"Time";TAB(30);time$
2200blue$=CHR$(132)+CHR$(157)+CHR$(134)
2210go$="A N O T H E R  G O    Y / N ?"
2220PRINT:PRINT
2230PRINTblue$;CHR$(141);CHR$(136);TAB(5);go$
2240PRINTblue$;CHR$(141);CHR$(136);TAB(5);go$
2250REM ** REPLY ? **
2260*FX 15,1
2270answer$=INKEY$(0)
2280IF GET$="N" THEN finish_flag=1
2290ENDPROC
2300:
2310REM **** LEVEL 2 PROCEDURES ****
2320DEF PROCinitialise_variables
2330xdet=2:ydet=25:xman=17:yman=1
2340ystart=120:xfinish=1144
2350zero$="000000"
2360ENDPROC
2370:
2380DEF PROCdefine_characters
2390REM ** MINES **
2400VDU23,224,0,0,56,254,254,124,0,0
2410REM ** MINE DETECTOR **
2420VDU23,225,231,195,189,36,36,189,195,231
2430REM ** ASSISTANT **
2440VDU23,226,56,56,16,124,186,170,40,108
2450ENDPROC
2460:
2470DEF PROCdraw_border
2480GCOL 0,1
2490MOVE 120,188
2500DRAW 120,992
2510DRAW 1152,992
2520DRAW 1152,188
2530DRAW 120,188
2540ENDPROC
2550:
2560DEF PROClay_mines(number_mines)
2570REM ** CHANGE COLOUR 2 TO GREEN **
2580VDU19,2,2,0,0,0
2590FOR I=1 TO number_mines
2600PRINTTAB(RND(16)+1,RND(25));CHR$(224)
2610NEXT I
2620ENDPROC
2630:
2640DEF PROCset_time
2650PRINTTAB(2,27)"Time     02:00"
2660TIME=0
2670ENDPROC
2680:
2690DEF PROCset_men
2700men$=CHR$(226)+CHR$(226)+CHR$(226)
2710count=1
2720COLOUR 1
2730PRINTTAB(2,30);men$
2740COLOUR 2
2750ENDPROC
2760:
2770DEF PROCset_score
2780score=0:score$="00000"
2790PRINTTAB(2,28)"Score    00000"
2800PRINTTAB(2,29)"Hi score ";hi_score$
2810ENDPROC
2820:
2830DEF PROCposition_chars
2840COLOUR 1
2850PRINTTAB(xdet,ydet);CHR$(225)
2860PRINTTAB(xman,yman);CHR$(226)
2870COLOUR 2
2880ENDPROC
2890:
2900DEF PROCupdate_time
2910sec$=STR$(((12100-TIME) DIV 100)MOD 60)
2920min$=STR$(((12100-TIME) DIV 6000)MOD 60)
2930REM ** ADD LEADING ZEROS **
2940sec$=LEFT$(zero$,2-LEN(sec$))+sec$
2950min$=LEFT$(zero$,2-LEN(min$))+min$
2960time$=min$+":"+sec$
2970PRINTTAB(11,27);time$
2980ENDPROC
2990:
3000DEF PROCtest_keyboard
3010 REM ** UP ? **
3020IF INKEY(-58)=-1 THEN PROCmove(0,-1)
3030REM ** DOWN ? **
3040IF INKEY(-42)=-1 THEN PROCmove(0,1)
3050REM ** RIGHT ? **
3060IF INKEY(-122)=-1 THEN PROCmove(1,0)
3070REM ** LEFT ? **
3080IF INKEY(-26)=-1 THEN PROCmove(-1,0)
3090ENDPROC
3100:
3110DEF PROCsnipe
3120ystart=RND(750)+220
3130yfinish=RND(750)+220
3140dx=32:dy=(yfinish-ystart)/32
3150GCOL 3,3
3160PROCline
3170IF POINT(x,y)=1 THEN PROCexplode(x,y) ELSE PROCline
3180ENDPROC
3190:
3200REM **** LEVEL 3 PROCEDURES ****
3210:
3220 DEF PROCmove(delta_x,delta_y)
3230REM ** RUB OUT OLD POSITIONS **
3240COLOUR 1
3250PRINTTAB(xdet,ydet);" "
3260PRINTTAB(xman,yman);" "
3270REM ** MOVE DETECTOR **
3280xdet=xdet+delta_x
3290ydet=ydet+delta_y
3300REM ** TEST FOR LIMITS **
3310IF xdet>17 THEN xdet=17
3320IF ydet>25 THEN ydet=25
3330IF xdet<2  THEN xdet=2
3340IF ydet<1  THEN ydet=1
3350REM ** CALCULATE MAN'S COORDS **
3360xman=19-xdet
3370yman=26-ydet
3380PROCconvert(xman,yman)
3390IF POINT(xgraph,ygraph)=2 THEN PROCexplode(xgraph,ygraph)
3400PROCconvert(xdet,ydet)
3410IF POINT(xgraph,ygraph)=2 THEN PROCfound_mine
3420PROCposition_chars
3430ENDPROC
3440:
3450DEF PROCline
3460SOUND0,-8,4,5
3470x=xstart:y=ystart
3480MOVE x,y
3490REPEAT
3500DRAW x,y
3510x=x+dx:y=y+dy
3520UNTIL x>xfinish OR POINT(x,y)=1
3530ENDPROC
3540:
3550DEF PROCexplode(x_explode,y_explode)
3560REM ** SOUND EFFECT **
3570SOUND 0,-15,6,50
3580REM ** SET FLASH RATE **
3590*FX9,20
3600*FX10,50
3610FOR I=1 TO 100
3620MOVE x_explode,y_explode
3630VDU19,2,RND(15),0,0,0
3640GCOL 0,RND(3)
3650PLOT 1,RND(100)-50,RND(100)-50
3660NEXT I
3670PROCreset
3680ENDPROC
3690:
3700REM **** LEVEL 4 PROCEDURES ****
3710:
3720 DEF PROCconvert(xchar,ychar)
3730xgraph=64*xchar+32
3740ygraph=1023-(32*ychar+16)
3750ENDPROC
3760:
3770DEF PROCfound_mine
3780REM ** SOUND EFFECT **
3790SOUND 2,-15,170,8
3800REM ** INCREMENT SCORE **
3810COLOUR 2
3820score=score+150
3830score$=STR$(score)
3840 score$=LEFT$(zero$,5-LEN(score$))+score$
3850PRINTTAB(11,28);score$
3860ENDPROC
3870:
3880DEF PROCreset
3890count=count+1
3900IF count>4 THEN end_flag=1:ENDPROC
3910CLS
3920VDU19,2,2,0,0,0
3930COLOUR 2
3940PROCinitialise_variables
3950mines_left=factor-score/150
3960PROClay_mines(mines_left)
3970PROCdraw_border
3980PRINTTAB(2,27);"Time"
3990PRINTTAB(2,28)"Score"
4000PRINTTAB(11,28)score$
4010PRINTTAB(2,29)"Hi score"
4020PRINTTAB(11,29)hi_score$
4030remaining_men$=LEFT$(men$,4-count)
4040COLOUR 1
4050PRINTTAB(2,30);remaining_men$;"   "
4060COLOUR 2
4070PROCposition_chars
4080ENDPROC
4090DEF PROCmusic
4100REM ** 1ST BAR **
4110SOUND1,-8,213,5
4120SOUND1,-8,209,5
4130SOUND1,-8,213,5
4140SOUND1,-8,209,5
4150SOUND1,-8,213,5
4160SOUND1,-8,193,5
4170SOUND1,-8,205,5
4180SOUND1,-8,197,5
4190REM ** 2ND BAR **
4200SOUND1,-8,185,20
4210SOUND1,-8,165,5
4220SOUND1,-8,185,5
4230SOUND1,-8,193,20
4240REM ** 3RD BAR **
4250SOUND1,-8,165,5
4260SOUND1,-8,193,5
4270SOUND1,-8,197,20
4280ENDPROC
```

MIDNIGHT RIDERS

Beyond Software's Lords of Midnight is claimed to be a new concept in computer games programming, because it combines elements from war games and adventures. The program's most spectacular claim to fame, however, is its use of an advanced graphics technique that gives literally thousands of different views of the action.

Computer war games, which are derived from board games such as Blitzkrieg and Diplomacy, differ considerably from adventure games, most of which are based, however loosely, on the classic Dungeons and Dragons. A war game requires the application of strategy and tactical planning, with all the pieces — armies, supplies, weapons, etc. — displayed on a map of the battleground. An adventure relies on surprise and resourcefulness — a player must solve a series of problems that are revealed one by one as progress is made through the scenario. In Lords of Midnight, for the 48 Kbyte Spectrum, Beyond Software has combined the two forms to produce a new type of game.

You are provided with a 30-page booklet, which contains a map of the Land of Midnight, in which the action is set. You play Luxor the Moonprince, Lord of the Free. Luxor has the Moon Ring, a device that allows him to control — and see through the eyes of — his four companions and any Lord of the Free that he can recruit. The Free are in control of most of the south, which must be defended against attack from Doomdark, the Witchking of Midnight. Doomdark's forces are controlled by the computer; based in a northern citadel, they attempt to take control of the southern areas.

Doomdark is aided by the Ice Crown, a magical device that casts the 'Ice Fear' into his enemies. This weakens any of the Free who approach the northern lands. However, one of your companions, Morkin, is immune to the Ice Fear, so his mission is to move stealthily northwards to destroy the Crown while the other characters are involved in openly fighting the forces of Doomdark.

Traditional war games are based on a map that allows players to keep track of the forces arrayed against them. In Lords of Midnight, this constantly updated map is not available to the players but is held in the Spectrum's memory. The view of the action that you have is through the eyes of a Lord of the Free, who can look in any one of eight directions, which are selected from the keyboard. So, although there may be an enemy army lying in wait beyond that range of hills ahead of you, you will be unable to see it until you go around the hills and look in the proper direction. This adds a new dimension to war gaming.

The most impressive feature of Lords of Midnight is the game's superb graphics. Beyond Software claims that 32,000 different scenes are available, and there are 32 characters with which to view them. Obviously, the Spectrum cannot hold this number of screens in memory, so Beyond has developed a technique called 'landscaping'. The thousands of different screen displays are all composed of 15 different shapes, each of which is available in four different sizes to indicate perspective. These basic shapes are combined to allow complex displays to be built up — hills, forests, mountains, armies, villages and citadels are all shown in detail. The Spectrum's memory holds a map giving the position of the elements in any one of 4,000 locations. This information allows the computer to calculate the view from any given point.

Lords of Midnight is beautifully conceived and presented. In keeping with the game's theme, Beyond has even redesigned the Spectrum character set to give a gothic flavour to text messages. What's more, unlike many other games that rely on beautiful graphics to hide a mundane plot, Lords of Midnight is gripping enough to keep you playing again and again.

Lords of Midnight: For the 48K Spectrum, £9.95
Publishers: Beyond Software, Competition House, Farndon Road, Market Harborough, Leicestershire LE19 9NR
Author: Mike Singleton
Joysticks: Not required
Format: Cassette

Gothic Effect
Beyond Software claim that there are 32,000 different scenes in the game, although it is doubtful whether anybody has attempted to count them! Note the Gothic lettering in the text. Beyond has redesigned the Spectrum character set to give the game a more 'teutonic' feel

LIZ HEANEY

Card Index
Lords of Midnight comes with a keyboard overlay card, making it easier for the player to find the correct command key

THE PLOT THICKENS

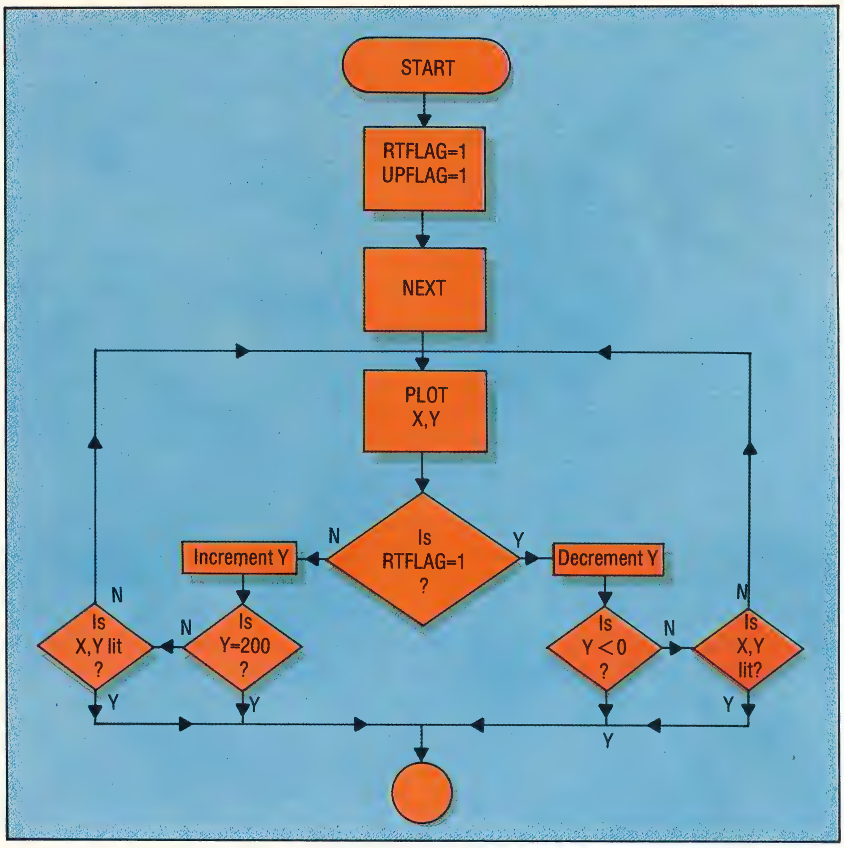
In the course so far, we have developed a series of machine code routines that utilise the high resolution capabilities of the Commodore 64. Plotsub (see page 337), Linesub (see page 416) and Circsub (see page 457) allow us to plot points and draw lines and circles, respectively. Here, we conclude the series with a routine that enables us to fill in the shapes drawn on the screen with the other programs.

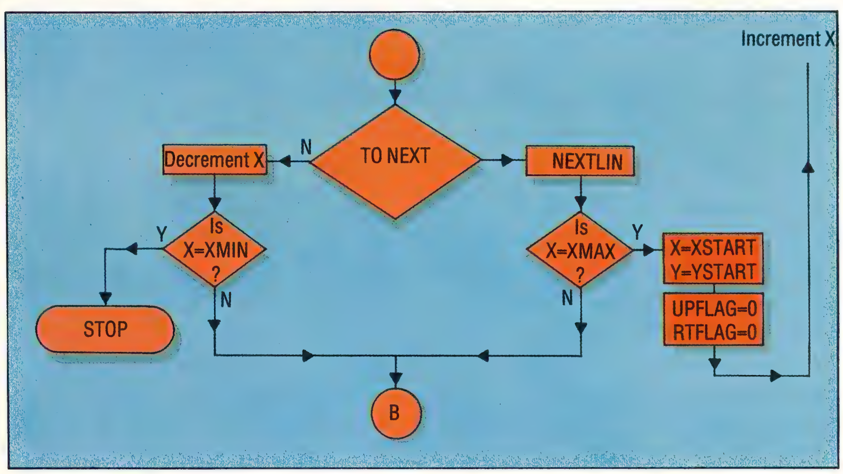
There are many ways of designing algorithms to fill shapes on a screen, but what, at first sight, seems to be a fairly simple task is in fact more complex than one might imagine. Shapes that contain 'reflex' angles (interior angles greater than 180 degrees) or shapes that contract and then expand all present their own particular difficulties. It is possible to design a single routine to cope with some of these difficulties but not all of them. Giving a program the 'intelligence' to assess what constitutes a closed shape is not an easy matter.

The method we shall use for our routine begins filling in the shape from a starting point, designated by the user, anywhere within the shape's outline. The routine then progresses up the screen, plotting points until it reaches a boundary, whereupon it moves right one pixel and progresses down the screen until it meets another boundary. Again, a one pixel move to the right is made and the routine starts moving up. This procedure continues until the right-hand side of the shape is filled. The whole process is then repeated, beginning again at the starting point and moving left, until the entire shape is filled.

The first part of the routine is straightforward. Two flags, UPFLAG and RTFLAG are used to indicate the direction in which the filling routine should move at any stage of the program. The first flowchart segment shows the plot-increment-test part of the routine. The main loop of this section increments or decrements the Y co-ordinate value depending on the state of UPFLAG. After testing for the edge of the screen and assessing whether the next pixel is already lit, the routine loops back to plot the next point. If a lit pixel or a screen edge is encountered, then the routine moves on to the next stage.

The routine will then move either right or left depending on the state of RTFLAG. Detecting the left or right boundary of the shape is difficult. Rather than doing this, the routine allows the user to set maximum and minimum values for the X co-ordinate. This facility also gives the user the option of filling in strips within a shape.

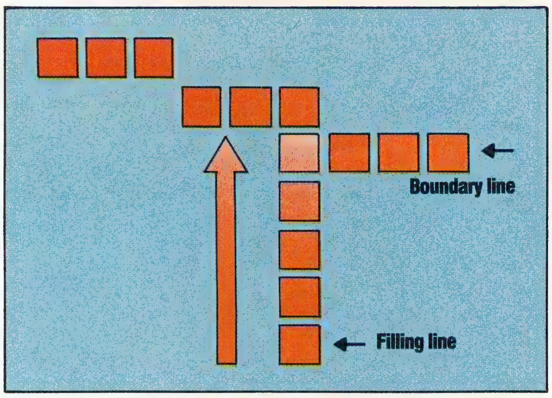




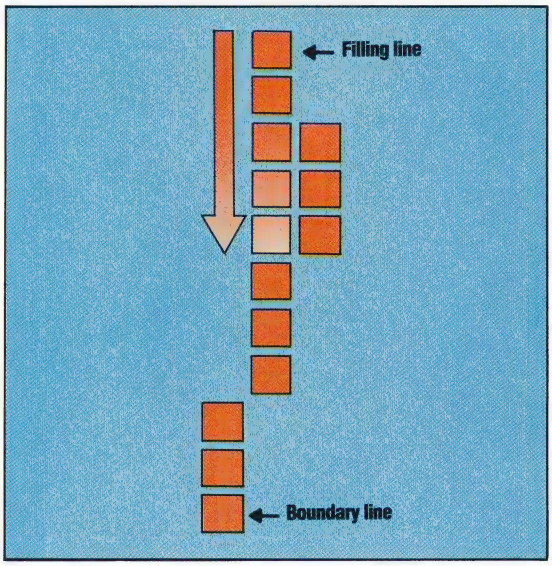
The second flowchart shows that if X is incrementing (i.e. moving right) and reaches its maximum value, then the values of X and Y are reset to the start co-ordinate values and the direction flags are both set to zero in preparation for filling the left part of the shape. If X is decrementing and reaches XMIN, then this indicates the completion of the routine. If the maximum or minimum values of X have not been reached, then the routine must fill in the next line.

PROBLEMS WITH GRADIENTS

The algorithm to cope with all the different possible types of line — shallow, steep, thick and thin — looks fairly complex, but the principles of its operation are relatively simple. Let us consider the situation where the fill routine is moving up the screen, and it encounters a line with a fairly shallow gradient.



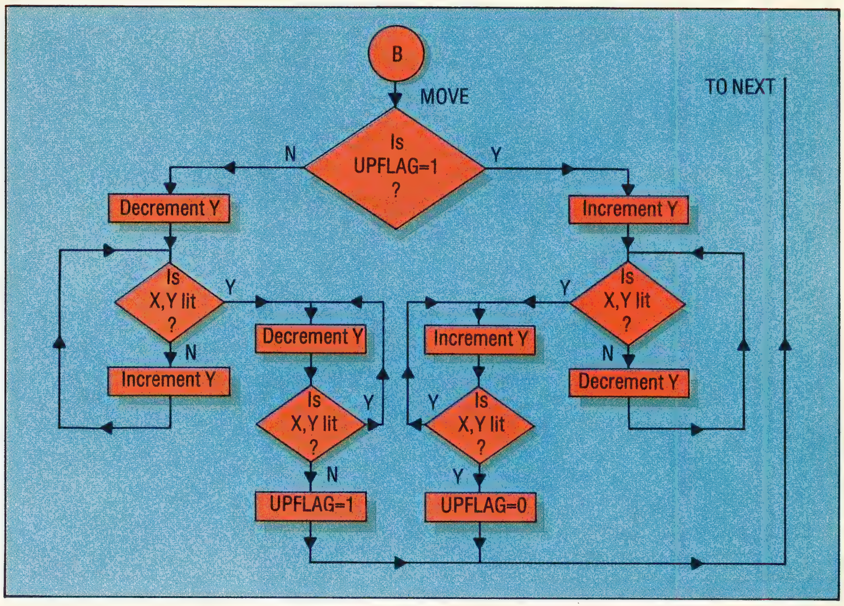
The first action that the algorithm takes on encountering the line is to backtrack one pixel. If the move is to the right then a further move needs to be made so that the next fill line can start at an unlit position. If we look at the case where a downward-moving fill line encounters a steep line we can see another problem to overcome.



If the fill is to move right then it must backtrack three pixels to find an empty space from which to start its next upward fill. If, however, the fill is to move left, then it must move another three pixels down the screen so that the new upward fill starts in the first unlit position above the boundary line. For each fill direction we must therefore have two loops to move in either direction until the first unlit position on the fill line is located. The flowchart showing this algorithm is given above.

THE FILLSUB LISTING

The main body of Fillsub follows the flowchart we have outlined fairly closely. Labels that have been used in the source code listing have been included

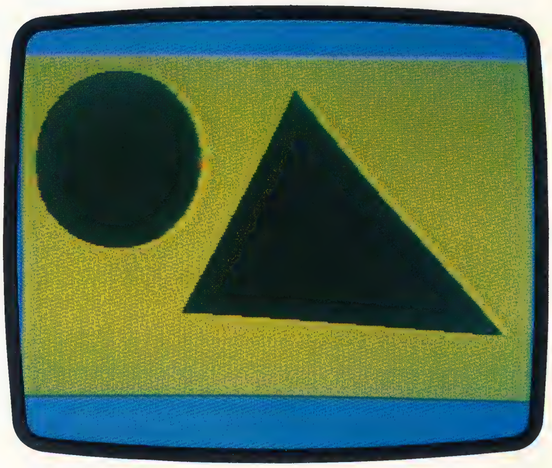


DIAGRAMS BY KEVIN JONES

in the corresponding positions on the flowchart to help you find your way around the program. In several places the fill algorithm requires us to test whether a point is lit or not. The program does this using a subroutine called POINT, which calculates the address of the point from its X,Y co-ordinates in the same way that our Plotsub program does. But instead of performing a logical OR to set a bit to one, POINT uses logical AND to see whether the bit in question is one or not. If the result of the AND operation is not zero, then the bit is a one. This result is stored in PTFLAG for later testing by the main program. Much of POINT does the same work as Plotsub and readers who feel comfortable with 6502 machine code may wish to modify Plotsub so that it can not only turn on a bit in the bitmap, but also test the value of any bit.

Finally, let's look at how the Fillsub routine is used. First of all, Fillsub needs several parameters to be passed to it. These are:

- The co-ordinates of the start point. This *must* be inside the shape we want to fill!
- The maximum and minimum X co-ordinate values to be used. Care must be taken when dealing with shapes that contain very acute angles, because Fillsub can move outside a shape if it reaches the edge of the interior space before



Shaping Up
After a number of articles, and having typed several kilobytes of code, we achieve finally on the Commodore 64 the kind of elementary hi-res graphic facilities that are taken for granted as standard BASIC commands on almost every other home micro on the market. The alternative to this kind of effort, of course, is to spend £30 to £50 on an extended BASIC cartridge

reaching one of the set limits. Resetting the limits slightly further in will overcome this problem. Note that the X co-ordinate of the start point and the limits of X must be split into hi-byte/lo-byte form, as shown in the demonstration program.

Although Fillsub does not directly rely on any of the other routines we have developed for the Commodore 64, the other three routines (Plotsub, Linesub and Circsub) are loaded by the demonstration program to draw the shapes to be filled by Fillsub.

One final point. When this routine was originally designed, the filling action took place in the horizontal direction rather than the vertical. But it was found that using vertical bars to fill the shape speeds up execution time considerably.

Plotsub/II Loader

This is an amended version of the Plotsub routine first published on page 339. Use it to create a new object file called "PLOTSUB.HEX" on cassette or disk, as explained on page 339

```
10 FORI=49408TO49408+314
20 READA:POKEI,A:S=S+A:NEXT
30 READCC:IFCC<>STHENPRINT"CHECKSUM ERROR"
100 DATA1,0,3,6,0,5,0,0,6,5,5,69,38,2
110 DATA72,138,72,152,72,173,0,193,240
120 DATA83,169,0,133,251,169,4,133,252
130 DATA162,3,160,0,173,2,193,145,251
140 DATA136,208,251,230,252,202,48,8
150 DATA208,244,145,251,160,231,208
160 DATA238,173,1,193,240,24,169,0,133
170 DATA251,169,32,133,252,162,32,160
180 DATA0,169,0,145,251,136,208,251
190 DATA230,252,202,208,246,173,24,208
200 DATA41,240,9,8,141,24,208,173,17
210 DATA208,9,32,141,17,208,76,125,193
220 DATA173,24,208,41,240,9,4,141,24
230 DATA208,173,17,208,41,223,141,17
240 DATA208,104,168,104,170,104,96,72
250 DATA138,72,152,72,173,4,193,141,7
260 DATA193,173,3,193,41,248,141,6,193
270 DATA173,3,193,41,7,141,8,193,173,5
280 DATA193,41,7,141,10,193,162,3,78,5
290 DATA193,202,208,250,173,5,193,141
300 DATA9,193,169,0,141,11,193,141,12
310 DATA193,162,5,173,11,193,24,109,9
320 DATA193,141,11,193,202,208,243,162
330 DATA6,14,12,193,14,11,193,144,3
340 DATA238,12,193,202,208,242,173,11
350 DATA193,24,109,6,193,141,11,193
360 DATA173,12,193,109,7,193,141,12
370 DATA193,173,11,193,24,105,0,141,11
380 DATA193,173,12,193,105,32,141,12
390 DATA193,173,11,193,24,109,10,193
400 DATA141,11,193,173,12,193,105,0
410 DATA141,12,193,173,11,193,133,251
420 DATA173,12,193,133,252,169,1,141
430 DATA13,193,56,169,7,237,8,193,240
440 DATA7,170,14,13,193,202,208,248
450 DATA160,0,177,251,13,13,193,145
460 DATA251,76,125,193
470 DATA37523:REM*CHECKSUM*
```

Strange Device

Line 15 DN=8 indicates that the object files (Plotsub.Hex, etc.) are to be loaded from disk. For tape use, change this to DN=1, and either make one tape with the object files in the order specified by lines 20 to 30 or, if your files are on different tapes, insert this code as lines 22,26 and 28:

INPUT"CHANGE TAPE & HIT RETURN";A$

Fillsub Demo

```
10 REM **** FILLSUB DEMO PROGRAM ****
15 DN=8:REM FOR CASSETTE DN=1
20 IFA=0THENA=1:LOAD"PLOTSUB.HEX",DN,1
25 IFA=1THENA=2:LOAD"LINESUB.HEX",DN,1
27 IFA=2THENA=3:LOAD"CIRCSUB.HEX",DN,1
30 IFA=3THENA=4:LOAD"FILLSUB.HEX",DN,1
40 GOSUB1000:REM SET HIRES
50 REM **** DRAW TRIANGLE ****
60 XA=100:YA=150:XB=300:YB=160:XC=170:YC=20
80 X1=XA:Y1=YA:X2=XB:Y2=YB:GOSUB2000
90 X1=XC:Y1=YC:GOSUB2000
100 X2=XA:Y2=YA:GOSUB2000
102 REM **** DRAW CIRCLE ****
103 XC=60:YC=60:R=50:GOSUB4000
120 REM **** FILL TRIANGLE ****
130 XS=170:YS=130:REM START POINTS
140 MIN=100:MAX=299:REM LIMITS
150 GOSUB3000
161 REM **** FILL CIRCLE ****
162 XS=60:YS=60:REM START POINT
163 MIN=10:MAX=109
164 GOSUB3000
```

Fillsub Demo (cont.)

```
200 GETA$:IFA$=""THEN200:REM AWAIT KEYPRESS
210 POKE49408,0:SYS49422:REM RESET SCREEN
220 PRINTCHR$(147):REM CLEAR SCREEN
225 PRINT"END OF ROUTINE"
230 END
1000 REM **** SET HIRES ****
1010 POKE49408,1:POKE49409,1
1020 POKE49410,7
1030 SYS49422
1040 RETURN
2000 REM **** LINESUB ****
2010 MHI=INT(X1/256):MLO=X1-256*MHI
2020 NHI=INT(X2/256):NLO=X2-256*NHI
2030 POKE49920,MLO:POKE49921,MHI
2040 POKE49922,NLO:POKE49923,NHI
2050 POKE49924,Y1:POKE49925,Y2
2060 SYS 49934
2070 RETURN
3000 REM **** FILLSUB ****
3010 SH=INT(XS/256):SL=XS-SH*256
3020 HAX=INT(MAX/256):LAX=MAX-256*HAX
3030 HIN=INT(MIN/256):LIN=MIN-256*HIN
3040 POKE50955,SL:POKE50956,SH
3050 POKE50957,YS
3060 POKE50958,LIN:POKE50959,HIN
3070 POKE50960,LAX:POKE50961,HAX
3080 SYS50967
3090 RETURN
4000 REM **** CIRCSUB ****
4010 CHI=INT(XXC/256):CLO=XC-256*CHI
4020 POKE50497,CLO:POKE50498,CHI
4030 POKE50499,YC:POKE50500,R
4040 SYS 50521
4060 RETURN
```

Fillsub Loader

```
10 REM **** BASIC LOADER FOR FILLSUB ****
20 FORI=50944 TO 51375
30 READA:POKEI,A:CC=CC+A:NEXT
40 READA:IFCC<>A THEN PRINT"CHECKSUM ERROR":END
100 DATA11,0,6,8,0,3,6,5,136,39,16,60
110 DATA0,60,10,0,109,0,0,1,10,0,16
120 DATA173,11,199,141,20,199,173,12
130 DATA199,141,21,199,172,13,199,169
140 DATA1,141,18,199,141,19,199,140,5
150 DATA193,173,20,199,141,3,193,173
160 DATA21,199,141,4,193,32,131,193
170 DATA173,19,199,208,8,200,192,200
180 DATA240,19,76,82,199,136,192,0,144
190 DATA11,32,246,199,173,22,199,208,3
200 DATA76,46,199,173,18,199,208,31
210 DATA173,20,199,56,233,1,141,20,199
220 DATA173,21,199,233,0,141,21,199
230 DATA205,15,199,208,65,173,20,199
240 DATA205,14,199,208,57,96,173,20
250 DATA199,24,105,1,141,20,199,173,21
260 DATA199,105,0,141,21,199,205,17
270 DATA199,208,34,173,20,199,205,16
280 DATA199,208,26,173,11,199,141,20
290 DATA199,173,12,199,141,21,199,172
300 DATA13,199,169,0,141,19,199,141,18
310 DATA199,76,46,199,173,19,199,208
320 DATA28,136,32,246,199,173,22,199
330 DATA208,4,200,76,191,199,238,19
340 DATA199,136,32,246,199,173,22,199
350 DATA208,247,76,46,199,200,32,246
360 DATA199,173,22,199,208,4,136,76
370 DATA219,199,206,19,199,200,32,246
380 DATA199,173,22,199,208,247,76,46
390 DATA199,72,138,72,152,72,140,2,199
400 DATA173,20,199,141,0,199,173,21
410 DATA199,141,1,199,173,1,199,141,4
420 DATA199,173,0,199,41,248,141,3,199
430 DATA173,0,199,41,7,141,5,199,173,2
440 DATA199,41,7,141,7,199,162,3,78,2
450 DATA199,202,208,250,173,2,199,141
460 DATA6,199,169,0,141,8,199,141,9
470 DATA199,162,5,173,8,199,24,109,6
480 DATA199,141,8,199,202,208,243,162
490 DATA6,14,8,199,46,9,199,202,208
500 DATA247,173,8,199,24,109,3,199,141
510 DATA8,199,173,9,199,109,4,199,141
520 DATA9,199,173,8,199,24,105,0,141,8
530 DATA199,173,9,199,105,32,141,9,199
540 DATA173,8,199,24,109,7,199,133,251
550 DATA173,9,199,105,0,133,252,169,1
560 DATA141,10,199,56,169,7,237,5,199
570 DATA240,7,170,14,10,199,202,208
580 DATA250,160,0,177,251,45,10,199
590 DATA141,22,199,104,168,104,170,104
600 DATA96
610 DATA50785:REM*CHECKSUM*
```

Assembly Listing

```
                ;
                ;++++++++++++++++++++
                ;++++++++++++++++++++
                ;++                ++
                ;++   FILLSUB 64   ++
                ;++                ++
                ;++++++++++++++++++++
                ;++++++++++++++++++++
                ;
                ;++++ PLOTSUB VALUES ++++
                ;
                PLTSUB = $C183
                XLO    = $C103
                XHI    = $C104
                YLO    = $C105
                MPBLO  = $00
                MPBHI  = $20
                PTR    = $FB
                *      = $C700
                ;
                ;++++ FILLSUB VARIABLES ++++
                ;
                PXLO   *=*+1
                PXHI   *=*+1
                PYLO   *=*+1
                PHBLO  *=*+1
                PHBHI  *=*+1
                PREMX  *=*+1
                PVBYTE *=*+1
                PREMY  *=*+1
                PROWLO *=*+1
                PROWHI *=*+1
                PBPOS  *=*+1
                ;
                XSTLO  *=*+1
                XSTHI  *=*+1
                YST    *=*+1
                XMINLO *=*+1
                XMINHI *=*+1
                XMAXLO *=*+1
                XMAXHI *=*+1
                RTFLAG *=*+1
                UPFLAG *=*+1
                FXLO   *=*+1
                FXHI   *=*+1
                PTFLAG *=*+1
                ;
                ;++++ TRANS START PT TO FX,FY ++++
                ;
AD 0B C7               LDA XSTLO
8D 14 C7               STA FXLO
AD 0C C7               LDA XSTHI
8D 15 C7               STA FXHI
AC 0D C7               LDY YST
                ;
                ;++++ SET FLAGS ++++
                ;
A9 01                  LDA #$01
8D 12 C7               STA RTFLAG
8D 13 C7               STA UPFLAG
                ;
                ;++++ PLOT POINT ++++
                ;
                NEXT
8C 05 C1               STY YLO
AD 14 C7               LDA FXLO
8D 03 C1               STA XLO
AD 15 C7               LDA FXHI
8D 04 C1               STA XHI
20 83 C1               JSR PLTSUB
                ;
                ;++++ INC / DEC Y COORD ++++
                ;
AD 13 C7               LDA UPFLAG
D0 08                  BNE DECRY
C8                     INY
C0 C8                  CPY #$C8          ;HAS Y REACHED MAX
F0 13                  BEQ NEXLIN
4C 52 C7               JMP TESTPT
                DECRY
88                     DEY
C0 00                  CPY #$00          ;HAS Y REACHED MIN
90 0B                  BCC NEXLIN
                TESTPT
20 F6 C7               JSR POINT         ;IS POINT LIT ?
AD 16 C7               LDA PTFLAG
D0 03                  BNE NEXLIN        ;IF YES BRANCH
4C 2E C7               JMP NEXT
                ;
                ;++++ START NEXT LINE ++++
                ;
                NEXLIN
AD 12 C7               LDA RTFLAG
D0 1F                  BNE INCRX         ; TEST RIGHT FLAG
AD 14 C7               LDA FXLO
38                     SEC
E9 01                  SBC #$01
8D 14 C7               STA FXLO          ;DEC X LOBYTE
AD 15 C7               LDA FXHI
E9 00                  SBC #$00
8D 15 C7               STA FXHI          ;DEC X HIBYTE
                ;
CD 0F C7               CMP XMINHI
D0 41                  BNE MOVE
AD 14 C7               LDA FXLO          ;HAS X REACHED XMIN ?
CD 0E C7               CMP XMINLO
D0 39                  BNE MOVE
60                     RTS               ;END OF ROUTINE
                ;
                INCRX
AD 14 C7               LDA FXLO
18                     CLC
69 01                  ADC #$01          ;INC X LOBYTE
8D 14 C7               STA FXLO
AD 15 C7               LDA FXHI
69 00                  ADC #$00          ;INC X HIBYTE
8D 15 C7               STA FXHI
                ;
CD 11 C7               CMP XMAXHI
D0 22                  BNE MOVE
AD 14 C7               LDA FXLO          ;HAS X REACHED XMAX ?
CD 10 C7               CMP XMAXLO
D0 1A                  BNE MOVE
                ;
AD 0B C7               LDA XSTLO
8D 14 C7               STA FXLO
AD 0C C7               LDA XSTHI         ;RESET START POINT
8D 15 C7               STA FXHI
AC 0D C7               LDY YST
                ;
A9 00                  LDA #$00
8D 13 C7               STA UPFLAG
8D 12 C7               STA RTFLAG        ;RESET FLAGS
4C 2E C7               JMP NEXT
                ;
                ;++++ FIND START OF NEXT LINE ++++
                ;
                MOVE
AD 13 C7               LDA UPFLAG
D0 1C                  BNE DOWN          ;TEST UPFLAG
88                     DEY               ;Y=Y-1
                AGAIN1
20 F6 C7               JSR POINT
AD 16 C7               LDA PTFLAG
D0 04                  BNE CONT1
C8                     INY
4C BF C7               JMP AGAIN1
                CONT1
EE 13 C7               INC UPFLAG        ;SET TO ONE
                AGAIN2
88                     DEY
20 F6 C7               JSR POINT
AD 16 C7               LDA PTFLAG
D0 F7                  BNE AGAIN2
4C 2E C7               JMP NEXT
                ;
                DOWN
C8                     INY               ;Y=Y+1
                AGAIN3
20 F6 C7               JSR POINT
AD 16 C7               LDA PTFLAG
D0 04                  BNE CONT2
88                     DEY
4C DB C7               JMP AGAIN3
                CONT2
CE 13 C7               DEC UPFLAG        ;SET TO ZERO
                AGAIN4
C8                     INY
20 F6 C7               JSR POINT
AD 16 C7               LDA PTFLAG
D0 F7                  BNE AGAIN4
4C 2E C7               JMP NEXT
                ;++++ END OF MAIN PROGRAM ++++
                ;
                ;++++ TEST POINT SUBROUTINE ++++
                ;
                POINT
48                     PHA
8A                     TXA
48                     PHA               ;PUSH REGS ONTO STACK
98                     TYA
48                     PHA
                ;
8C 02 C7               STY PYLO
AD 14 C7               LDA FXLO
8D 00 C7               STA PXLO;TRANSFER COORDS
AD 15 C7               LDA FXHI
8D 01 C7               STA PXHI
                ;
                ;++++ CALCULATE ADDRESS OF POINT ++++
                ;
AD 01 C7               LDA PXHI
8D 04 C7               STA PHBHI
AD 00 C7               LDA PXLO
29 F8                  AND #$F8
8D 03 C7               STA PHBLO
AD 00 C7               LDA PXLO
29 07                  AND #$07
8D 05 C7               STA PREMX
                ;
AD 02 C7               LDA PYLO
29 07                  AND #$07
8D 07 C7               STA PREMY
                ;
A2 03                  LDX #$03
                SHIFT
4E 02 C7               LSR PYLO
CA                     DEX
D0 FA                  BNE SHIFT
AD 02 C7               LDA PYLO
8D 06 C7               STA PVBYTE
                ;
A9 00                  LDA #$00
8D 08 C7               STA PROWLO
8D 09 C7               STA PROWHI
A2 05                  LDX #$05
                FIVE
AD 08 C7               LDA PROWLO
18                     CLC
6D 06 C7               ADC PVBYTE
8D 08 C7               STA PROWLO
CA                     DEX
D0 F3                  BNE FIVE
A2 06                  LDX #$06
                MULT
0E 08 C7               ASL PROWLO
2E 09 C7               ROL PROWHI
CA                     DEX
D0 F7                  BNE MULT
                ;
AD 08 C7               LDA PROWLO
18                     CLC
6D 03 C7               ADC PHBLO
8D 08 C7               STA PROWLO
AD 09 C7               LDA PROWHI
6D 04 C7               ADC PHBHI
8D 09 C7               STA PROWHI
                ;
AD 08 C7               LDA PROWLO
18                     CLC
69 00                  ADC #MPBLO
8D 08 C7               STA PROWLO
AD 09 C7               LDA PROWHI
69 20                  ADC #MPBHI
8D 09 C7               STA PROWHI
                ;
AD 08 C7               LDA PROWLO
18                     CLC
6D 07 C7               ADC PREMY
85 FB                  STA PTR
AD 09 C7               LDA PROWHI
69 00                  ADC #$00
85 FC                  STA PTR+1
                ;
A9 01                  LDA #$01
8D 0A C7               STA PBPOS
38                     SEC
A9 07                  LDA #$07
ED 05 C7               SBC PREMX
F0 07                  BEQ BITON
AA                     TAX
                POWER
0E 0A C7               ASL PBPOS
CA                     DEX
D0.FA                  BNE POWER
                ;
                ;++++ TEST FOR BIT ON ++++
                ;
                BITON
A0 00                  LDY #$00
B1 FB                  LDA (PTR),Y       ;LOAD ADDRESS CONTENTS
2D 0A C7               AND PBPOS
8D 16 C7               STA PTFLAG        ;STORE RESULT

68                     PLA
A8                     TAY
68                     PLA               ;PULL REGS OFF STACK
AA                     TAX
68                     PLA

60                     RTS
```

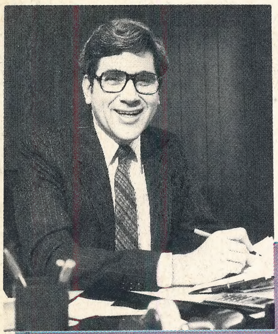
QUALITY CONTROL

Softsel is the world's largest wholesale distributor of computer hardware and software. If you have ever used a business package or a game on a home computer from an American-based company, then it is very likely that the software was distributed through this company.

Softsel provides computer software retailers with an extremely valuable service. At a time when software packages, some of them very expensive, are being released at an ever-increasing rate, a dealer is faced with the problem of evaluating each new product personally, which is time-consuming and costly, or else trusting to a hurried evaluation, which could mean being stuck with some unsaleable and expensive packages. The service that Softsel provides is to remove this element of risk for the retailer, by performing extensive product evaluation on all the packages that the company adds to its software list.

Simon Rhodes, UK marketing manager for the company, explains the procedure: 'The package is first examined by our technical department for user-friendliness — checking whether it is well-programmed, well-documented, has good graphics, and so on. It is then passed to our marketing and sales department who decide whether it will receive good promotion and advertising.' Softsel estimates that in a recent six month period, only 10 per cent of nearly 700 packages were accepted onto the company's catalogue.

This quality control, coupled with the added incentives of sale and return agreements and the need to deal with only one supplier, make a company like Softsel an attractive proposition for a software retailer. Undoubtedly, a retailer could buy the packages at a cheaper price direct from a manufacturer, but Softsel's ability to offer discounts through its bulk-buying power will make the difference only marginal. Simon Rhodes points out: 'Although a dealer could get a better price, in the long run it would cost more to buy direct from the manufacturers, because he would have to deal with hundreds of different people rather than just one company.'



Herb Blumstein, Managing Director

Simon Rhodes, Marketing Services Manager

IAN McKINNELL

AMERICAN ORIGINS

Softsel was founded in 1980 by Robert Leff and David Wagman, who had been co-workers in the data processing department of Transaction Technology, a subsidiary of the giant Citicorp finance house. Leff and Wagman's belief in the need for a company such as Softsel was vindicated by its rapid expansion. Four years after it was established, Softsel employs 350 staff worldwide, and the company's international turnover in the last full trading year amounted to a massive $87 million. In the US alone, the company has four large warehouses — in Atlanta, Chicago, Los Angeles and New York — supplying a range of 4,500 packages to dealers across the country.

The company's penetration of the UK market began in September 1982. A subsidiary, Softsel Computer Products, was established in April of the following year. The UK branch is based in Feltham, close to Heathrow Airport, and supplies over 2,500 different products to dealers all over Europe and the Middle East.

The future of the company looks bright. The UK subsidiary plans to increase the percentage of its software catalogue devoted to business packages, which already take up more than half of the catalogue. However, the company does not expect to play down the importance of the other major area of software production — games programs.

Softsel also intends to increase its share of the European market. A subsidiary has already been established in Germany, with its main office in Munich, and there are plans for French and Italian subsidiaries to be set up in the autumn of 1984.

On page 101 of The Home Computer Advanced Course we introduced you to Micronet, an area within Prestel (British Telecom's public viewdata service) that is designated for the use of microcomputer owners. Micronet allows thousands of home computer users the opportunity to:

- Look at up-to-the-minute news;
- Download free or chargeable software;
- Send and receive messages to and from other subscribers to the Prestel system.

Anyone wishing to become a member of Micronet must first buy a modem to connect their computer to the telephone line. This can cost as little as £70. Then there are Prestel's subscription charges of £13 per quarter, and on top of that the cost of the phone calls themselves.

Since our previous article was written there have been some major changes to the organisation of the microcomputing databases on Prestel. Instead of subscribing direct to Micronet, micro owners must now join 'Prestel Microcomputing'. This is an umbrella name for several databases, of which the main one is still Micronet 800. The other two are Viewfax 258 and ClubSpot 810. Both Micronet and Viewfax are commercial areas, selling telesoftware and providing news and computer interest sections. However, the third database, ClubSpot, as its name implies, is run by members of computer clubs.

ClubSpot began several years ago, before Micronet came into existence. It began as a small part of one of Prestel's experimental areas, and contained information about the Association of London Computer Clubs. As time went on, it proved a popular success and moved on to the Practical Computing pages of Prestel, followed by the IPC Practical Telesoftware pages. Last year, when Micronet was set up, ClubSpot became a sub-Information Provider within Micronet, with its front page at 8008. At that point it had grown to over 1,500 frames of information. It is now on the move again, as Prestel have just given ClubSpot the facilities of a Main Information Provider, and the front page for this is 810.

The major concern of ClubSpot is to cater to computer clubs and their members. A major benefit for computer clubs is in recruiting new members, and many clubs find their pages on ClubSpot useful for displaying club news and information for their members. Several computer clubs use their ClubSpot pages as the focal point of club meetings. The pages are updated with information and club news almost immediately. For example, the North London Computer Club has a complete list of all the clubs and courses it organises, as well as club news.

ClubSpot also includes information produced by enthusiasts, containing material of interest to ordinary Micronet members. There are pages concerned with shows and exhibitions, communication systems other than Prestel, a schools area, and several examples of hobbyist initiatives, such as BeebSpot, Adventure Helpline and the Art Gallery. BeebSpot is produced by two 15-year-olds, Michael Sparrow and Richard Ryczanowski, and it provides BBC Micro owners with news and information about all aspects of the machine.

One of the most entertaining uses of computers is in playing adventure games and it will be of no surprise to find out that the Adventure Helpline is one of the most popular areas in ClubSpot. As well as many pages of reviews of adventure games, and hints and tips for playing specific adventures there are also 15 'Helpliners' who do their best to deal with over 25 queries a week from frustrated adventurers.

But ClubSpot's pride and joy is the Art Gallery. This is the brainchild of Dr David Annal, who has a talent for getting the best possible graphics results from the low-resolution graphics used on Prestel. His impressive artwork is produced on a Commodore Pet before being placed on the system. He is also responsible for many of the well-designed frames seen elsewhere in ClubSpot and Prestel.

IMAGES PRODUCED ON THE ARTRON 2000 STUDIO COMPUTER; COURTESY OF GRAPHIC PRODUCTS, LONDON (EXCLUSIVE DISTRIBUTOR)